Giorgio Magri (IJN, ENS)
magrigrg@gmail.com

EALing 2010, ENS
Paris, September 8, 2010

The OT online model of the acquisition of phonotactics

Class 2: converge of cautious promotion, and invariants

Summary — This class introduces the goal of devising promotion/demotion OT update rules that converge for any input OT compatible tableau; introduces a new family of OT update rules, that perform promotion, but cautiously "not too much"; shows that OT-compatibility of the data entails conic independence of the update vectors used by OT online algorithms; uses this fact to prove convergence for the proposed family of cautious promotion/demotion OT update rules; introduces the use of invariants to characterize the behavior of OT online algorithms. Thus, the main point of this class is that convergent constraint promotion is possible, as far as we cautiously do not "promote too much".

1 The core research question

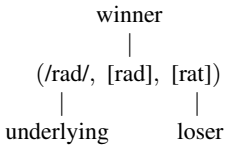
■ **Online models of the early stage.** My modeling task is the *early stage* of the acquisition of phonotactics, characterized by the following input/output properties:

(1) a. *Properties of the input.* Throughout this stage, the child does not yet have knowledge of morphology and is thus blind to alternations.

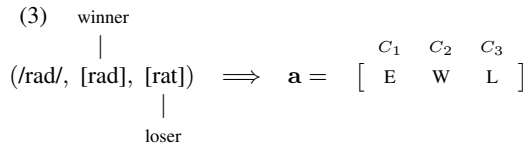
b. *Properties of the output.* By the end of this stage, the child is able to distinguish legal from illegal sequences w.r.t. his target language.

My modeling tool is online algorithms within the framework of OT. Let me quickly review the main properties of OT online algorithms.

■ **First property.** At each time, OT online algorithms take a current *data triplet* consisting of an underlying form, an intended winner form and a loser form, as (2).

(2)


A data triplet can be represented as a *comparative row* $\mathbf{a}$ that specifies whether a constraint is *winner-preferring* (WPC) or *loser-preferring* (LPC) or even, as in (3).

(3)


C_1 = IDENT[VOICE]/ONSET
C_2 = IDENT[VOICE]
C_3 = [VOICE]

Many data triplets correspond to many comparative rows that can be stacked one on top of the other into a *comparative tableau*.

■ **Second property.** OT online algorithms entertain a current *OT grammar* at every time, represented as a *ranking vector* $\boldsymbol{\theta}$, namely an n-tuple of numerical *ranking values* $\theta_1, \ldots, \theta_n$, one for every constraint.

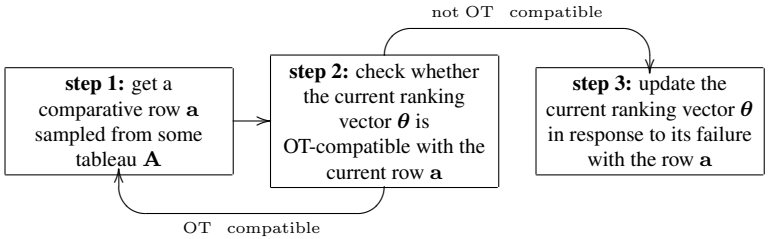
(4) $\boldsymbol{\theta} = \overset{C_1 \quad \ldots \quad C_k \quad \ldots \quad C_n}{(\ \theta_1, \quad \ldots \quad \theta_k, \quad \ldots \quad \theta_n\ )}.$

A ranking vector *represents* a ranking $\gg$ iff (5) holds for any pair of constraints C_h, C_k, namely $\gg$ respects the ordering implicit in the relative size of the ranking values.

(5) The ranking value θ_h is larger than the ranking value $\theta_k \Longrightarrow C_h \gg C_k$

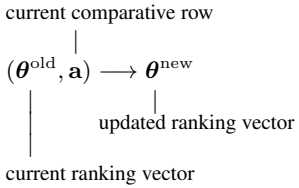
A ranking vector is *OT-compatible* with a comparative tableau iff all the rankings it represents are OT-compatible with it.

■ **Third property.** OT online algorithms maintain a current ranking vector. Initialize it to a pre-defined initial ranking vector. And keep updating it through the three steps (6).

(6)


Rows fed to the algorithm at step 1 are sampled from a fixed OT-compatible comparative tableau $\mathbf{A}$, called the *input tableau*.

■ **Research question.** OT online algorithms differ for the OT update rule (7) they use in step 3. An *OT update rule* has the following form:

(7)


Thus, the core research question is: what is an OT update rule that yields a proper model of the early stage of the acquisition of phonotactics?

2 We need constraint promotion

■ **Demotion-only vs promotion/demotion.** OT update rules come in two varieties:

(8)

	demote LPCs	promote WPCs
Demotion-only update rules:	√	
Promotion-demotion update rules	√	√

Class 1 presented the beautiful example (9) of demotion-only update rules, from Tesar and Smolensky (1998) and Boersma (1997):

$$(9)\quad \theta_k^{\text{new}} = \begin{cases} \theta_k^{\text{old}} - 1 & \text{if } C_k \text{ is an undominated LPC} \\ \theta_k^{\text{old}} & \text{otherwise} \end{cases}$$

as well as example (10) of promotion/demotion update rules, from Boersma (1997):

$$(10)\quad \theta_k^{\text{new}} = \begin{cases} \theta_k^{\text{old}} - 1 & \text{if } C_k \text{ is a LPC} \\ \theta_k^{\text{old}} + 1 & \text{if } C_k \text{ is a WPC} \\ \theta_k^{\text{old}} & \text{if } C_k \text{ is an even constraint} \end{cases}$$

■ **We need constraint promotion.** Suppose we try to model the early stage of the acquisition of phonotactics with an OT online algorithm that performs demotion only. Then:

(11) a. during the early stage, the learner has no access to alternations;
b. thus, the model posits fully faithful underlying forms;
c. thus, the faithfulness constraints are never LPCs;
d. thus, the faithfulness constraints are never re-ranked by demotion-only;

And this cannot be right, as seen in class 1. Thus:

(12) The OT online model of the early stage cannot work with demotion only; rather, we need constraint promotion in order to re-rank the faithfulness constraints too, despite the fact that they are always WPCs.

But how should we choose a proper promotion/demotion update rule?

3 We want universally convergent constraint promotion

■ **Convergence.** The OT online algorithm (6) *converges* on an input comparative tableau $\mathbf{A}$ iff it is not possible to construct an infinite sequence $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_t, \ldots$ of rows of $\mathbf{A}$ such that each row $\mathbf{a}_t$ in the sequence triggers an update if fed to the algorithm at time t:

(13) $\boldsymbol{\theta}^{\text{init}} \xrightarrow{\mathbf{a}_1} \boldsymbol{\theta}_1 \xrightarrow{\mathbf{a}_2} \boldsymbol{\theta}_2 \xrightarrow{\mathbf{a}_3} \cdots \longrightarrow \boldsymbol{\theta}_t \xrightarrow{\mathbf{a}_{t+1}} \cdots$

$\boldsymbol{\theta}^{\text{init}}$ is not OT-comp. with $\mathbf{a}_1$ and thus update is triggered

$\boldsymbol{\theta}_1$ is not OT-comp. with $\mathbf{a}_2$ and thus update is triggered

$\boldsymbol{\theta}_2$ is not OT-comp. with $\mathbf{a}_3$ and thus update is triggered

$\cdots$

$\boldsymbol{\theta}_t$ is not OT-comp. with $\mathbf{a}_{t+1}$ and thus update is triggered

To illustrate, recall from class 1 that the OT online algorithm:

(14) a. does *indeed* converge for *any* OT-compatible input tableau, in the case of the demotion-only update rule (9);
b. does *not* converge for *every* OT-compatible input tableau, in the case of the promotion/demotion update rule (10): Pater's (2008) counterexample.

We want convergent update rules. But convergent for which tableaux? Two options:

(15) a. require convergence for *any* OT-compatible input tableau | we take convergence to follow from the general ranking logic of OT

b. require convergence for a *special* subset of OT-compatible input tableaux | we take convergence to follow from peculiar properties of this special family of tableaux

I will call (15a) *universal convergence*. Which option should we pick, (15a) or (15b)?

■ **Ranking problem.** The simplest problem in computational OT is (16).

(16) *given:* an OT-compatible OT-comparative tableau $\mathbf{A}$;
find: a ranking $\gg$ OT-compatible with the tableau $\mathbf{A}$.

I denote by RP($\mathbf{A}$) the instance of the *Ranking problem* (16) for a tableau $\mathbf{A}$.

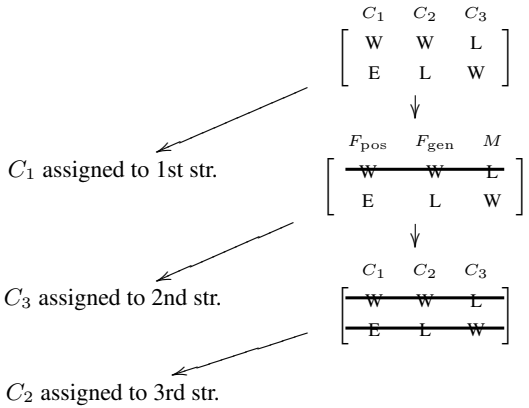
■ **Claim 1** The Ranking problem RP($\mathbf{A}$) is easily solvable (in time linear in the number of columns of $\mathbf{A}$), *for any OT-compatible tableau* $\mathbf{A}$.

Proof. T&S note that the RP($\mathbf{A}$) can be solved by repeating the two steps (17).

(17) a. Assign to the currently available top stratum, a constraint that has no L's in the current tableau;
b. update the current tableau by deleting all rows where the currently ranked constraint has a W.

An example of the procedure (17) is provided in (18): diagonal arrows in (18) correspond to step (17a) and vertical arrows to step (17b).

(18)



Algorithm (17) is *Recursive Constraint Demotion*. It is a restatement from an algorithmic perspective of T&S's characterization of OT-compatibility (claim 1, class 1). □

■ **We want universally convergent update rules.** I now reason as follows:

(19) a. The OT online algorithm converges on an input tableau $\mathbf{A}$ iff it solves the corresponding Ranking problem RP($\mathbf{A}$);
b. if the Ranking problem RP($\mathbf{A}$) cannot be solved for all tableaux $\mathbf{A}$, the we would have to "settle' for option (15b) of non-universal convergence;
c. but claim 1 ensures that the Ranking problem RP($\mathbf{A}$) can indeed be solved for *any* (OT compatible) tableaux $\mathbf{A}$;
d. thus let's be bold, and try to go for universal convergence (15a), investigating whether the bare logic of OT is able to ensure convergence.

This is an instance of the modeling strategy (20) outlined in class 1; we will see another example in class 4.

(20) a. Single out the building blocks of the learning task (e.g. the Ranking problem);
b. study their complexity (e.g. claim 1);
c. devise algorithms that solve these problems up to their complexity class.

Thus, this class has two main goals:

(21) a. develop a family of promotion/demotion OT update rules that are provably universally convergent;
b. provide some preliminary support for the heuristics of requiring universal convergence, by taking a preliminary look at correctness.

4 Cautious constraint promotion

■ **Constraint promotion is hard to get.** T&S warn against constraint promotion:

(22) "Couldn't the learner just as easily promote constraints toward the correct hierarchy? The answer is no" Tesar and Smolensky (1998, pp. 244-245).

The only promotion-demotion update rule considered in the literature is Boersma's (10). Yet, Pater (2008) shows that:

(23) There are cases where the OT online algorithm run with Boersma's update rule (10) performs an infinite number of updates and thus never converges.

Thus, constraint promotion has so far eluded the efforts of the computational OT literature. Is it possible to devise convergent promotion/demotion OT update rules?

■ **First step.** Suppose that the current comparative row has a unique L and a unique W:

(24)
	C_h		C_ℓ	
[...	W	...	L	...]

I submit the following intuition:

(25) a. The comparative row (24) unambiguously says that the WPC C_h must be ranked above the LPC C_ℓ in the end;
b. thus, in this case we can be *bold*;
c. namely, we can promote by the same amount we demote, say 1.

Thus, I suggest to update as follows, namely the same as Boersma's update rule (10):

(26) a. Demote the unique LPC by 1;
b. promote the unique WPC by 1.

■ **Second step.** Suppose the current comparative tableau has a unique L but multiple W's:

(27)
[...	W	W	W	...	L	...]

Boersma's update rule (10) is not *cautious*, because it overall promotes too much, namely more (i.e. $1 + 1 + 1 = 3$) than it demotes (i.e. 1). Indeed, recall from class 1 that the ranking values kept growing in the case of Pater's counterexample.

(28)
[...	W	W	W	...	L	...]
	add 1	add 1	add 1		subtract 1	

3 times

I submit the following intuition:

(29) a. This comparative row (27) does *not* unambiguously say which one of the multiple WPCs needs in the end be ranked above the LPC;
b. thus, we should *not* be bold and promote each WPC by the same amount we demote the unique LPC, as in the preceding case;
c. rather, we should be *cautious* and split that amount over the multiple WPCs.

In other words, I suggest the following *cautious* update:

(30)
[...	W	W	W	...	L	...]
	add $\frac{1}{3}$	add $\frac{1}{3}$	add $\frac{1}{3}$		subtract 1	

3 times

More in general, I suggest the following *cautious* update:

(31) a. Demote the unique LPC by 1;
b. promote each WPC by $\frac{1}{\text{total number of WPCs}}$.

Of course, if I multiply (31) by the total number of WPCs, nothing changes (OT update rules are invariant by positive rescaling — provided the initial ranking vector has identical components). Thus, I can describe (31) as follows:

(32) a. Demote the unique LPC by the total number of WPCs;

b. Promote each WPC by 1.

■ **Third step.** Suppose now the current comparative tableau has multiple *undominated* L's and multiple W's:

(33)

	C_h	C_k		C'	C''	
...	W	W	...	L	L	...

This comparative row (33) is *equivalent* to (i.e. OT-compatible with the same rankings as) the two comparative rows (34): the two L's of (33) can be split over two rows.

(34) a.

	C_h	C_k		C'	C''	
...	W	W	...	L		...

b.

	C_h	C_k		C'	C''	
...	W	W	...		L	...

I thus submit the following intuition:

(35) a. Update triggered by the original row (33) should be equivalent to two consecutive updates triggered by the two new rows (34a) and (34b).

b. Update by the first row (34a) can be performed according to (32), since it has a unique LPC: each WPC is promoted by 1 and the unique LPC is demoted by the total number of WPCs.

c. Subsequent update by the second row (34b) can be performed again according to (32): each WPC is promoted again by 1 and the unique LPC is demoted by the total number of WPCs.

d. In conclusion, each WPC gets promoted by the total number of LPCs and each LPC gets demoted by the total number of WPCs.

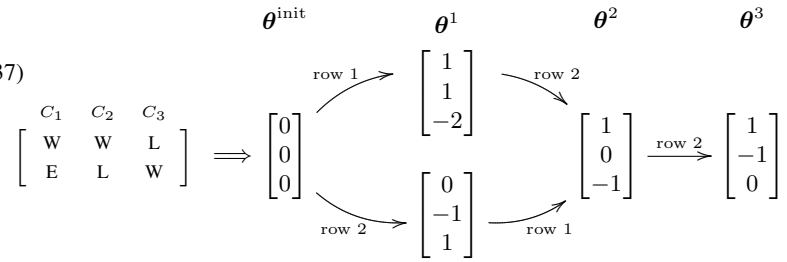
Thus, I suggest the following *cautious* update:

(36) a. Demote all undominated LPCs by the total number of WPCs;

b. promote all WPCs by the total number of undominated LPCs.

■ **An example.** The behavior of the OT online algorithm (6) with the new cautious promotion-demotion update rule (36) is illustrated in the diagram (37).

(37)



Note the non-monotonic dynamics of the ranking values (they go up and down), which is the hallmark of update rules that perform promotion too.

5 Cautious promotion works!

■ **Claim 2** The OT online algorithm (6) run with the new cautious promotion-demotion update rule (36) converges for *any* OT-compatible input tableau.

■ **Generalization.** Let me say that a promotion-demotion update rule is *cautious* iff it does not "promote too much", in the following sense:

(38) For any comparative row, the sum of the amounts by which the WPCs are promoted is smaller than or at most equal to the sum of the amounts by which the undominated LPCs are demoted.

Constraint promotion is delicate. T&S are drastic: they suggest that we do not promote at all. It turns out we can be a bit less drastic: we can promote, but cautiously. In fact, convergence actually holds for any cautious update rule (provided it only demotes undominated LPCs and that all WPCs are promoted).

■ **The actual parameterization.** In class 1, we saw that OT-grammars can be parameterized equivalently by *rankings* and *ranking vectors*. The following question thus arises:

(39) Do speakers assume that OT languages are parameterized by *combinatoric* objects such as rankings or by *continuous* objects such as ranking vectors?

The computational perspective adopted here offers a handle on question (39) — other perspectives being modeling gradualness, gradiance, etc. In fact:

(40) a. *Demotion-only can be implemented with rankings:*
for example, T&S's original demotion-only OT online algorithm represents the current grammar by means of rankings;

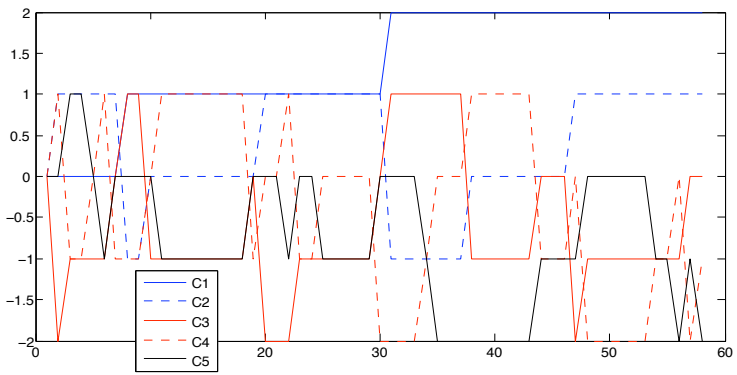
b. *Promotion/demotion can only be implemented with ranking vectors:*
as we have just seen, it is crucial to promote by the proper promotion amount; hence, the rule is inherently numerical and it thus requires ranking vectors.

If indeed promotion is needed to model, say, the early stage of the acquisition of phonotactics, then the answer to (39) is: *continuous ranking vectors*, not discrete rankings.

6 Sketch of the proof of convergence

■ **Preliminaries.** An example of the dynamics of the current ranking values entertained by the OT online algorithm (6) with the cautious promotion/demotion update rule (36) run on Pater's comparative tableau (see class 1) with the rows sampled uniformly is (41).

(41)



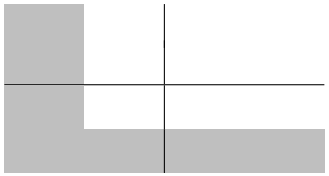
This dynamics is complicated and non-monotonic, contrary to the case of demotion-only update rules. Thus, the proof of convergence is more involved. It has four steps.

■ **First step.** The following fact holds:

(42) The ranking values entertained by OT online algorithms with any update rule (that only demotes *undominated* LPCs) cannot get smaller than a given constant.

In other words, the current ranking vector cannot live in the shaded region of (43).

(43)



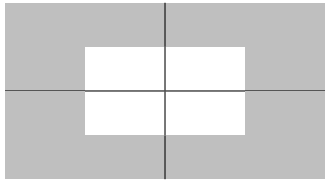
Proof. As seen in class 1, T&S show that (42) holds for update rules that perform demotion only (of undominated LPCs). But a close look at their proof reveals that it holds for pretty much any update rule (that demotes only undominated LPCs). □

■ **Second step.** The following fact holds:

(44) The ranking values entertained by OT online algorithms with a *cautious* promotion-demotion update rule cannot get larger than a given constant.

Together with (42), this means that the current ranking vector entertained by the algorithm cannot live in the shaded region in (45).

(45)



Proof. Let me focus on the specific cautious promotion-demotion update rule (36). The ranking vectors entertained by the algorithm in the run (37) are repeated in (46).

(46) $\begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$

Note that the components of each ranking vector in (46) sum up to zero. This is a general property of the new cautious promotion-demotion update rule (36):

(47) The sum of the current ranking values at any given time is zero.

In fact, suppose that the sum $\sum_{k=1}^{n} \theta_k^{\text{old}}$ of the current rankings values is zero; suppose that the current comparative tableau has w WPCs and ℓ undominated LPCs (ULPCs). Then, the sum $\sum_{k=1}^{n} \theta_k^{\text{new}}$ of the updated ranking values is zero too:

(48)
$$\sum_{k=1}^{n} \theta_k^{\text{new}} \overset{(a)}{=} \sum_{k\,2\,\text{WPC}} \theta_k^{\text{new}} + \sum_{k\,2\,\text{ULPC}} \theta_k^{\text{new}} + \sum_{\text{remaining } k} \theta_k^{\text{new}}$$

split the set $\{1, \ldots, n\}$ that k runs over into the set WPC of winner-preferrers, the set ULPC of undominated loser-preferrers and their complement

$$\overset{(b)}{=} \sum_{k\,2\,\text{WPC}} \left(\theta_k^{\text{old}} + \ell\right) + \sum_{k\,2\,\text{ULPC}} \left(\theta_k^{\text{old}} - w\right) + \sum_{\text{remaining } k} \theta_k^{\text{old}}$$

use (36) to express the updated ranking values θ_k^{new} in terms of the current ranking values θ_k^{old}

$$\overset{(c)}{=} \sum_{k=1}^{n} \theta_k^{\text{old}} + \sum_{k\,2\,\text{WPC}} \ell - \sum_{k\,2\,\text{ULPC}} w$$

by reordering the terms

$$\overset{(d)}{=} \sum_{k\,2\,\text{WPC}} \ell - \sum_{k\,2\,\text{ULPC}} w$$

by the hypothesis $\sum_{k=1}^{n} \theta_k^{\text{old}} = 0$ that the current ranking values sum to zero

$$= w\ell - \ell w = 0$$

I can now conclude the reasoning as follows:

(49)
a. since the current ranking values cannot become too small, by (42),
b. since furthermore they must always add up to zero, by (47),
c. then they cannot become too large either, as stated in (44).

For an arbitrary initial ranking vector and an arbitrary cautious update rule, the sum of the current ranking values is always smaller than a fixed constant (instead than equal to zero); the reasoning in (49) thus carries over. □

■ **Third step.** The following fact holds:

(50) The OT online algorithm with any update rule can never loop, namely it can never entertain again a current ranking vector that it had previously dismissed (as long as all the WPCs are promoted and the input tableau is OT-compatible).

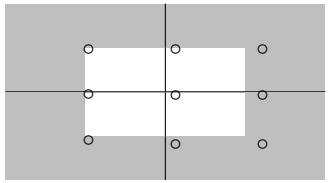
In other words, the OT online algorithm can never walk through a sequence of ranking vectors as (51), whereby the algorithm loops back to $\boldsymbol{\theta}$ after having dismissed it for $\boldsymbol{\theta}^0$.

(51) $\ldots \longrightarrow \boldsymbol{\theta} \longrightarrow \ldots \longrightarrow \boldsymbol{\theta}^0 \longrightarrow \ldots \longrightarrow \boldsymbol{\theta} \longrightarrow \ldots$

Proof. This is the non-trivial step; I will come back to this in the next section. □

■ **Fourth step.** Since the ranking vectors entertained by the algorithm must live on a lattice and must furthermore live within a bounded region by (42) and (44), then the search space of the algorithm is finite, as illustrated in (52).

(52)



Since the algorithm cannot loop by (50), finiteness of the search space entails finite time convergence.

7 The third step of the proof, in some detail

■ **Conic combinations.** Consider ℓ vectors $\overline{\mathbf{a}}_1, \ldots, \overline{\mathbf{a}}_\ell$ with the same number n of components. Consider a combination of these vectors with some coefficients $\alpha_1, \ldots, \alpha_\ell$:

$$\alpha_1\overline{\mathbf{a}}_1 + \ldots \alpha_\ell\overline{\mathbf{a}}_\ell = \alpha_1 \begin{bmatrix} \overline{a}_1^1 \\ \vdots \\ \overline{a}_n^1 \end{bmatrix} + \ldots + \alpha_\ell \begin{bmatrix} \overline{a}_1^\ell \\ \vdots \\ \overline{a}_n^\ell \end{bmatrix} \tag{53}$$

Here is a little example:

$$2 \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}^{\overline{\mathbf{a}}_1} + 4 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}^{\overline{\mathbf{a}}_2} + 3 \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}^{\overline{\mathbf{a}}_3} = \begin{bmatrix} 6 \\ -1 \\ 3 \end{bmatrix} \tag{54}$$

If the coefficients $\alpha_1, \ldots, \alpha_\ell$ are all non-negative, then (53) is a *conic combination*.

■ **Conic independence.** The following two properties (55) and (56) are equivalent.

(55) A conic combination of $\overline{\mathbf{a}}_1, \ldots, \overline{\mathbf{a}}_\ell$ adds up to the null vector iff the coefficients $\alpha_1, \ldots, \alpha_\ell$ are all null:

$$\left.\begin{aligned} \alpha_1 &\geq 0 \\ &\vdots \\ \alpha_\ell &\geq 0 \\ \alpha_1\overline{\mathbf{a}}_1 + \ldots \alpha_\ell\overline{\mathbf{a}}_\ell &= \mathbf{0} \end{aligned}\right\} \Longrightarrow \left\{\begin{aligned} \alpha_1 &= 0 \\ &\vdots \\ \alpha_\ell &= 0 \end{aligned}\right.$$

(56) Two conic combinations of $\overline{\mathbf{a}}_1, \ldots, \overline{\mathbf{a}}_\ell$ add up to the same vector despite the fact that the coefficients $\alpha_1, \ldots, \alpha_\ell$ of one of the two are not smaller than the coefficients $\alpha_1^0, \ldots, \alpha_\ell^0$ of the latter iff the coefficients are all identical:

$$\left.\begin{aligned} \alpha_1 &\geq \alpha_1^0 \\ &\vdots \\ \alpha_\ell &\geq \alpha_\ell^0 \\ \alpha_1\overline{\mathbf{a}}_1 + \ldots \alpha_\ell\overline{\mathbf{a}}_\ell &= \alpha_1^0\overline{\mathbf{a}}_1 + \ldots \alpha_\ell^0\overline{\mathbf{a}}_\ell \end{aligned}\right\} \Longrightarrow \left\{\begin{aligned} \alpha_1 &= \alpha_1^0, \\ &\vdots \\ \alpha_\ell &= \alpha_\ell^0 \end{aligned}\right.$$

Let me show for instance that property (55) entails (56):

(57)
$$\begin{aligned} &\alpha_1\overline{\mathbf{a}}_1 + \ldots \alpha_\ell\overline{\mathbf{a}}_\ell = \alpha_1^0\overline{\mathbf{a}}_1 + \ldots \alpha_\ell^0\overline{\mathbf{a}}_\ell \Longrightarrow \\ &\overset{(a)}{\Longrightarrow} \quad (\alpha_1 - \alpha_1^0)\overline{\mathbf{a}}_1 + \ldots + (\alpha_\ell - \alpha_\ell^0)\overline{\mathbf{a}}_\ell = \mathbf{0} \\ &\qquad \text{by reordering} \\ &\overset{(b)}{\Longrightarrow} \quad \left\{\begin{aligned} (\alpha_1 - \alpha_1^0) &= 0 \\ &\vdots \\ (\alpha_\ell - \alpha_\ell^0) &= 0 \end{aligned}\right. \\ &\qquad \text{by property (55)} \\ &\overset{(c)}{\Longrightarrow} \quad \left\{\begin{aligned} \alpha_1 &= \alpha_1^0 \\ &\vdots \\ \alpha_\ell &= \alpha_\ell^0 \end{aligned}\right. \end{aligned}$$

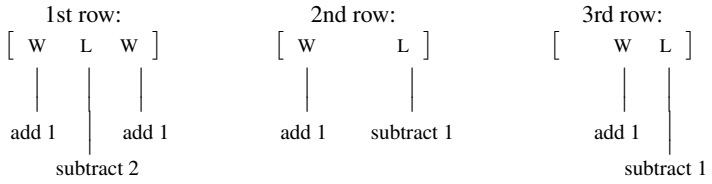
If either (55) or (56) holds, then $\overline{\mathbf{a}}_1, \ldots, \overline{\mathbf{a}}_\ell$ are called *conically independent*.

■ **OT compatibility entails conic independence.** To warm up, consider the following simple OT-compatible comparative tableau:

(58)
$$\begin{array}{c} \\ \text{row 1} \\ \text{row 2} \\ \text{row 3} \end{array} \begin{array}{c} \begin{array}{ccc} C_1 & C_2 & C_3 \end{array} \\ \begin{bmatrix} \text{W} & \text{L} & \text{W} \\ \text{W} & & \text{L} \\ & \text{W} & \text{L} \end{bmatrix} \end{array}$$

Update triggered by the three rows of (58) according to (36) can be described as follows:

(59)



Equivalently, update triggered by the three rows of (58) according to (36) can be described as adding to the current ranking vector one of the vectors in (68), called the corresponding *update vector*:

(60)

1st row: $\overline{\mathbf{a}}_1 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$ 2nd row: $\overline{\mathbf{a}}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ 3rd row: $\overline{\mathbf{a}}_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$

Consider a combination of the vectors in (68) that adds up to the null vector. Since the first entry in each vector is positive or null, then the coefficients α_1 and α_2 corresponding to vectors with non-null first entries needs to be 0. And

As noted above, update triggered by one of the three rows of (67) according to (36) consists of adding to the current ranking vector the corresponding update vector in (68).

(68) 1st row: $\overline{\mathbf{a}}_1 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$ 2nd row: $\overline{\mathbf{a}}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ 3rd row: $\overline{\mathbf{a}}_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$

Furthermore, consider the three coefficients $\alpha_1^t, \alpha_2^t, \alpha_3^t$ as follows:

(69)

α_1^t	α_2^t	α_3^t
$\|\|$	$\|\|$	$\|\|$
# of updates up to time t triggered by the 1st row of tableau (67)	# of updates up to time t triggered by the 2nd row of tableau (67)	# of updates up to time t triggered by the 3rd row of tableau (67)

Then, the ranking vector $\boldsymbol{\theta}^t$ entertained by the algorithm at time t can be described as follows (we already saw an analogous description, in the explanation of Pater's counterexample in class 1).

(70)

$$\boldsymbol{\theta}^t = \begin{bmatrix} \theta_1^t \\ \theta_2^t \\ \theta_3^t \end{bmatrix} = \alpha_1^t \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} + \alpha_2^t \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + \alpha_3^t \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

α_1^t: # of updates in response to the 1st row of the tableau (67)

α_3^t: # of updates in response to the 3rd row of the tableau (67)

$\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$: update vector that corresponds to the 1st row of the tableau (67) according to rule (36)

$\begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$: update vector that corresponds to the 3rd row of the tableau (67) according to rule (36)

Suppose now that the algorithm entertains the same ranking vector $\boldsymbol{\theta}^{t'} = \boldsymbol{\theta}^{t''}$ at time t^0 and at some later time t^{00}. This means that:

(71) $$\underbrace{\alpha_1^{t'} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} + \alpha_2^{t'} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + \alpha_3^{t'} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}}_{\boldsymbol{\theta}^{t'}} = \underbrace{\alpha_1^{t''} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} + \alpha_2^{t''} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + \alpha_3^{t''} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}}_{\boldsymbol{\theta}^{t''}}$$

Recall that the update vectors are conically independent. Since $\alpha_i^{t'} \leq \alpha_i^{t''}$ (as time t^{00} follows t^0), characterization (56) of conic independence thus entails (72).

(72) $\alpha_1^{t'} = \alpha_1^{t''}, \quad \alpha_2^{t'} = \alpha_2^{t''}, \quad \alpha_3^{t'} = \alpha_3^{t''}$

These identities (72) say that not a single update has happened between times t^0 and t^{00}. Hence, the algorithm could not have entertained a different ranking vector in between these two times. The reasoning straightforwardly extends to the general case.

■ **Remark.** No matter the details, I think there is something exciting in the reasoning just presented, that can be brought out as follows:

(73)
a. No matter the framework, many online algorithms work by adding to a current vector a certain update vector, see Cesa-Bianchi and Lugosi (2006);
b. thus, it is a general property of online algorithms that the current vector is a conic combination of a finite set of update vectors;
c. thus, it is important to study properties of conic combinations of the update vectors, of their conic geometry;
d. among all possible conic properties, one of the most important ones is of course conic independence (55);
e. by optimal design, we might thus want to assume a framework which ensures conic independence of the update vectors;
f. Voilá: OT is such a model!

This reasoning contributes to the recent debate on the comparison of OT with alternative models; see Bane and Riggle (2010), Tesar (2007) and Pater (2009) a.o. We will come back to this in class 3.

8 An invariant for promotion/demotion OT online algorithms

■ **Invariants for demotion-only.** As seen in class 1, T&S's analysis of the OT online algorithm with the beautiful demotion-only update rule (9) includes the following *invariant* on the current ranking values:

(74) If the input tableau is OT compatible with the ranking $C_1 \gg C_2 \gg \ldots \gg C_n$, then the current ranking value of constraint C_k can never drop below $-(k-1)$.

In general, invariants for demotion-only OT algorithms bound the ranking value θ_k of the kth constraint in a given run of the OT online algorithm, i.e. have the following form:

(75) If the input tableau is such and such, then $\theta_k \geq bound$.

■ **Invariants for promotion-demotion.** Of course, we want invariants also for promotion/demotion OT online algorithms. Yet:

(76)
a. in the promotion/demotion case, the dynamics of the ranking values θ_k is very complicated because non-monotonic, as in picture (41);
b. thus, we bound instead the number of times α_i that the ith row triggers an update, whose dynamics is very simple (α_i can only increase);
c. and then we use the relationship (70) between the ranking values θ_k and the coefficients α_i to derive invariants on the current ranking values.

Thus, I want to bound the number of updates α_i triggered by the ith row $\mathbf{a}_i$ in a given run of the OT online algorithm. To simplify the discussion, I will assume that:

(77) a. the initial ranking vector has all identical components;

b. the input tableau has a unique L per row.

By (77a), at least one update by $\mathbf{a}_i$ is necessary. Our question thus becomes (78): how many further updates will $\mathbf{a}_i$ trigger in the worst-case?

(78) If the input tableau is such and such, then $\alpha_i - 1 \leq ??$

I will concentrate on the cautious promotion-demotion update rule (36), repeated in (79);

(79) a. Promote every WPC by 1;
b. demote the unique LPC by the total number of WPCs.

Yet, the reasoning extends to any promotion-demotion update rule and does not require assumptions (77), made only to simplify the presentation.

■ **General shape of the bound.** Intuitively, we might reason as follows:

(80) a. If the input tableau only consisted of row $\mathbf{a}_i$, one update by row $\mathbf{a}_i$ would be enough to ensure that the current ranking vector is OT compatible with $\mathbf{a}_i$;
b. in order for $\mathbf{a}_i$ to trigger another update, that ranking configuration needs to be disrupted by update by some *disruptor row* $\mathbf{a}_j$ of the input tableau;
c. it might then take a certain number of further updates by row $\mathbf{a}_i$ in order to re-establish a ranking configuration OT-compatible with $\mathbf{a}_i$;
d. that number of further updates by row $\mathbf{a}_i$ will depend on the *amount of disruption* caused by the disruptor row $\mathbf{a}_j$.

Thus, let:

(81) a. $disruptor(\mathbf{a}_i)$ = the set of rows of the input tableau that trigger an update that disrupts the ranking configuration necessary for OT compatibility with row $\mathbf{a}_i$;
b. $disruption_j$ = the amount of disruption caused by a disruptor row $\mathbf{a}_j$ to the ranking configuration necessary for OT compatibility with row $\mathbf{a}_i$.

By (80), we expect the number $\alpha_i - 1$ of further updates triggered by row $\mathbf{a}_i$ to be bound by the sum over the numbers α_j of updates triggered by disruptor rows $\mathbf{a}_j$, each multiplied by the corresponding amount of disruption:

(82) $$\alpha_i - 1 \leq \sum_{j 2 disruptor(\mathbf{a}_i)} disruption_j \cdot \alpha_j$$

To make (82) explicit, we need to make (81) explicit, namely to define disruptor rows and their amount of disruption.

■ **Disruptors.** Suppose the target row $\mathbf{a}_i$ has an L corresponding to C_ℓ and a W corresponding to some C_h, as in (83). A row $\mathbf{a}_j$ can be a disruptor of the ranking configuration necessary for OT-compatibility with row $\mathbf{a}_i$ in one of three ways.

(83)

	...	C_h	...	C_ℓ	...	
$\mathbf{a}_i$	...	W	...	L	...	$\Longrightarrow$ { pushes up C_h; pushes down C_ℓ

First, row $\mathbf{a}_j$ can be an *L-disruptor*, namely has an L corresponding to constraint C_h, so that $\mathbf{a}_j$ pushes down the constraint C_h that $\mathbf{a}_i$ pushes up.

(84)

	...	C_h	...	C_ℓ	...	
$\mathbf{a}_j$	...	L	...	...	...	$\Longrightarrow$ { pushes down C_h

Second, row $\mathbf{a}_j$ can be a *W-disruptor*, namely has a W corresponding to constraint C_ℓ, so that $\mathbf{a}_j$ pushes up the constraint C_ℓ that $\mathbf{a}_i$ pushes down.

(85)

	...	C_h	...	C_ℓ	...	
$\mathbf{a}_j$	...	...	...	W	...	$\Longrightarrow$ { pushes up C_ℓ

Third, row $\mathbf{a}_j$ can be an *LW-disruptor*, namely has an L corresponding to C_h *and* a W corresponding to C_ℓ, so that $\mathbf{a}_j$ pushes down the constraint C_h that $\mathbf{a}_i$ pushes up and pushes up the constraint C_ℓ that $\mathbf{a}_i$ pushes down.

(86)

	...	C_h	...	C_ℓ	...	
$\mathbf{a}_j$	...	L	...	W	...	$\Longrightarrow$ { pushes down C_h; pushes up C_ℓ

Thus, the set of disruptors $disruptor_h(\mathbf{a}_i)$ of row $\mathbf{a}_i$ (relative to its WPC C_h) is the collection of all rows of the input tableau of type (84), (85) and (86).

■ **Disruption.** If row $\mathbf{a}_j$ is an L-disruptor of the target row $\mathbf{a}_i$ as in (84), then:

(87) a. update by row $\mathbf{a}_j$ according to (79) demotes C_h by the total number of WPCs of row $\mathbf{a}_j$, despite the fact that $\mathbf{a}_i$ would have promoted it;
b. thus, $\mathbf{a}_j$ disrupts the ranking configuration necessary for compatibility with $\mathbf{a}_i$ by an amount $disruption_j$ proportional to the number of WPCs of $\mathbf{a}_j$.

If row $\mathbf{a}_j$ is a W-disruptor of the target row $\mathbf{a}_i$ as in (85), then:

(88) a. update by row $\mathbf{a}_j$ according to (79) promotes C_ℓ by 1, despite the fact that $\mathbf{a}_i$ would have demoted it;
b. thus, $\mathbf{a}_j$ disrupts the ranking configuration necessary for compatibility with $\mathbf{a}_i$ by an amount $disruption_j$ proportional to 1.

If row $\mathbf{a}_j$ is an LW-disruptor of the target row $\mathbf{a}_i$ as in (86), then:

(89) a. update by $\mathbf{a}_j$ according to (79) promotes C_ℓ by 1 *and* demotes C_h by the number of WPCs of $\mathbf{a}_j$, despite the fact that $\mathbf{a}_i$ would have done the reverse;
b. thus, $\mathbf{a}_j$ disrupts the ranking config. necessary for compatibility with $\mathbf{a}_i$ by an amount $disruption_j$ proportional to 1 *plus* the number of WPCs of $\mathbf{a}_j$.

Let me summarize (87)-(89) as follows:

(90) $$disruption_j \sim \begin{cases} w(\mathbf{a}_j) & \text{if row } \mathbf{a}_j \text{ is an L-disruptor, as in (84)} \\ 1 & \text{if row } \mathbf{a}_j \text{ is a W-disruptor, as in (85)} \\ 1 + w(\mathbf{a}_j) & \text{if row } \mathbf{a}_j \text{ is an LW-disruptor, as in (86)} \end{cases}$$

where $w(\mathbf{a}_j)$ is the total number of WPCs in row $\mathbf{a}_j$.

■ **More on disruption.** Given the target row $\mathbf{a}_i$ in (83), we expect the amount of disruption $disruption_j$ caused by a disruptor row $\mathbf{a}_j$ to depend also on the target row $\mathbf{a}_i$:

(91)
a. Update by the target row $\mathbf{a}_i$ in (83) according to (79) promotes the WPC C_h by 1 and demotes the unique LPC C_ℓ by the number of WPCs of row $\mathbf{a}_i$;
b. thus, the *separation* between the ranking value of the LPC C_ℓ and the ranking value of the WPCs will increase by 1 plus the number of WPCs of $\mathbf{a}_i$;
c. if row $\mathbf{a}_i$ has many WPCs, then the separation between WPCs and the LPC after one update by $\mathbf{a}_i$ will increase a lot and will not be easily disrupted;
d. if row $\mathbf{a}_i$ has few WPCs, then the separation between WPCs and the LPC after one update by $\mathbf{a}_i$ will increase only a little and will be easily disrupted.

These considerations suggest that the amount of disruption $disruption_j$ should also be inversely proportional to 1 plus the total number of WPCs of the target row $\mathbf{a}_i$:

(92) $$disruption_i \sim \frac{1}{w(\mathbf{a}_i) + 1}$$

where $w(\mathbf{a}_i)$ is the total number of WPCs in row $\mathbf{a}_i$.

9 The invariant actually holds!

■ **Claim 3** Consider a row $\mathbf{a}_i$ of an OT-compatible input tableau that has a unique L per row. Let C_h be a WPC of row $\mathbf{a}_i$. The number of updates α_i^t triggered by $\mathbf{a}_i$ at time t in a run of the OT online algorithm with the promotion/demotion update rule (79) starting from the initial null ranking vector can be bound as follows:

(93) $$\alpha_i^t - 1 \leq \sum_{\mathbf{a}_j \,2\, disruptor_h(\mathbf{a}_i)} disruption_j \cdot \alpha_j^t$$

where the disruption amounts $disruption_j$ are defined as follows:

(94) $$disruption_j = \begin{cases} \dfrac{w(\mathbf{a}_j)}{w(\mathbf{a}_i) + 1} & \text{if row } \mathbf{a}_j \text{ is an L-disruptor, as in (84)} \\ \dfrac{1}{w(\mathbf{a}_i) + 1} & \text{if row } \mathbf{a}_j \text{ is an W-disruptor, as in (85)} \\ \dfrac{w(\mathbf{a}_j) + 1}{w(\mathbf{a}_i) + 1} & \text{if row } \mathbf{a}_j \text{ is an LW-disruptor, as in (86)} \end{cases}$$

Proof. The claim obviously holds if $\alpha_i^t = 1$. Thus, assume that $\alpha_i^t \geq 2$. Let s be the time preceding t when the row $\mathbf{a}_i$ has triggered the last update:

(95)
a. $\alpha_i^t = \alpha_i^s + 1$;
b. the ranking vector $\boldsymbol{\theta}^s = (\theta_1^s, \ldots, \theta_n^s)$ at time s is not OT-compatible with $\mathbf{a}_i$.

Let C_ℓ be the unique LPC of row $\mathbf{a}_i$; let C_h be any WPC of row $\mathbf{a}_i$. Condition (95b) thus entails in particular (96b).

(96)
a. $\alpha_i^t = \alpha_i^s + 1$
b. $\theta_h^s \leq \theta_k^s$

By reasoning as in (70), the ranking value θ_h^s of C_h at time s can be expressed as follows:

(97) $$\theta_h^s = \underbrace{\alpha_i^s}_{\substack{\text{contribution of} \\ \text{row } \mathbf{a}_i}} + \underbrace{\sum_{fj \neq i\, j\, C_h\, 2 \mathrm{WPC}(\mathbf{a}_j) g} \alpha_j^s}_{\substack{\text{contribution of rows } \mathbf{a}_j \\ \text{different from } \mathbf{a}_i \text{ where } C_h \\ \text{is a WPC}}} - \underbrace{\sum_{fj \neq i\, j\, C_h\, 2 \mathrm{LPC}(\mathbf{a}_j) g} w(\mathbf{a}_j)\alpha_j^s}_{\substack{\text{contribution of rows } \mathbf{a}_j \\ \text{different from } \mathbf{a}_i \text{ where } C_h \\ \text{is a LPC}}}$$

And analogously for the ranking value θ_ℓ^s of the LPC C_ℓ at time s:

(98) $$\theta_\ell^s = -w(\mathbf{a}_i)\alpha_i^s + \sum_{fj \neq i\, j\, C_\ell\, 2 \mathrm{WPC}(\mathbf{a}_j) g} \alpha_j^s - \sum_{fj \neq i\, j\, C_\ell\, 2 \mathrm{LPC}(\mathbf{a}_j) g} w(\mathbf{a}_j)\alpha_j^s$$

Using expressions (97)-(98), inequality (96b) becomes:

(99) $$\begin{aligned} &\alpha_i^s + \sum_{fj \neq i\, j\, C_h\, 2 \mathrm{WPC}(\mathbf{a}_j) g} \alpha_j^s - \sum_{fj \neq i\, j\, C_h\, 2 \mathrm{LPC}(\mathbf{a}_j) g} w(\mathbf{a}_j)\alpha_j^s \leq \\ &\leq -w(\mathbf{a}_i)\alpha_i^s + \sum_{fj \neq i\, j\, C_\ell\, 2 \mathrm{WPC}(\mathbf{a}_j) g} \alpha_j^s - \sum_{fj \neq i\, j\, C_\ell\, 2 \mathrm{LPC}(\mathbf{a}_j) g} w(\mathbf{a}_j)\alpha_j^s \end{aligned}$$

I make the left hand side of (99) smaller by dropping positive terms and the right hand side larger by dropping negative terms:

(100) $$\alpha_i^s - \sum_{fj \neq i\, j\, C_h\, 2 \mathrm{LPC}(\mathbf{a}_j) g} w(\mathbf{a}_j)\alpha_j^s \leq -w(\mathbf{a}_i)\alpha_i^s + \sum_{fj \neq i\, j\, C_\ell\, 2 \mathrm{WPC}(\mathbf{a}_j) g} \alpha_j^s$$

By trivially rearranging the terms of the inequality (100), I get:

(101) $$\alpha_i^s \leq \sum_{fj \neq i\, j\, C_\ell\, 2 \mathrm{WPC}(\mathbf{a}_j) g} \frac{1}{1 + w(\mathbf{a}_i)} \alpha_j^s + \sum_{fj \neq i\, j\, C_h\, 2 \mathrm{LPC}(\mathbf{a}_j) g} \frac{w(\mathbf{a}_j)}{1 + w(\mathbf{a}_i)} \alpha_j^s$$

The hypothesis that $s < t$ entails that $\alpha_i^s \leq \alpha_i^t$. I can therefore weaken (101) by replacing α_j^s with α_j^t on the right hand side:[1]

(102) $$\alpha_i^s \leq \sum_{fj \neq i\, j\, C_\ell\, 2 \mathrm{WPC}(\mathbf{a}_j) g} \frac{1}{1 + w(\mathbf{a}_i)} \alpha_j^t + \sum_{fj \neq i\, j\, C_h\, 2 \mathrm{LPC}(\mathbf{a}_j) g} \frac{w(\mathbf{a}_j)}{1 + w(\mathbf{a}_i)} \alpha_j^t$$

The claim follows by replacing $\alpha_s^i = \alpha_i^t - 1$ from (96) in the left hand side of (102). □

[1] This step is licit only because all the α's in the right hand side of (101) are multiplied by a positive coefficient, as a result of the simplification from (99) to (100).

10 A little application

■ **A test case.** Prince and Tesar (2004) collect benchmark test cases for models of the early stage of the acquisition of phonotactics. One of their test cases is:

(103)
	F_1	F_2	M_1	M_2
row 1	W		L	
row 2		W	W	L

Given as input the tableau in (103), the algorithm needs to recognize that F_2 is useless to OT-compatibility and thus should be ranked at the bottom as in (104), in order to ensure restrictiveness — we will extensively come back to this issue in class 4.

(104) $F_1 \gg M_1 \gg M_2 \gg F_2$

How does the OT online algorithm perform on this test case? Assume a biased initial ranking vector $\boldsymbol{\theta}^{\text{init}}$ as (105), that ranks the two markedness constraints well above the two faithfulness constraints.

(105) $$\begin{aligned} \theta^{\text{ini}}_{F_1} = \theta^{\text{ini}}_{F_2} &= 0 \\ \theta^{\text{ini}}_{M_1} = \theta^{\text{ini}}_{M_1} &= \theta^{\text{init}} = \text{a large positive constant} \end{aligned}$$

We need constraint promotion to pull apart F_1 and F_2. Thus, consider a promotion/demotion update rule (106). With $\lambda = 1$, this is Boersma's update rule (10); with $\lambda =$ # of WPCs, this is the cautious update rule (36), as rows 1 and 2 have a unique L.

(106) a. Promote all WPCs by 1;

b. demote the LPCs by a demotion amounts λ (λ can depend on the row considered).

Does the OT online algorithm run on input tableau (103) starting from the initial ranking vector (105) with the update rule (106) converge to the desired final ranking (104)?

■ **Intuition.** The analysis has two steps. The *first step* is (107), corresponding to claim 4.

(107) The algorithm converges to the desired final ranking (104) provided the total number α_2^{fin} of updates triggered by row 2 is "small".

The intuition behind (107) is straightforward:

(108) F_2 is only promoted by row 2 and thus the algorithm will keep F_2 low provided row 2 triggers few updates.

The *second step* of the analysis is (109).

(109) The total number α_2^{fin} of updates triggered by row 2 is "small" (independently of the frequency with which the row is fed to the algorithm) provided the demotion amount λ_2 in the case of update triggered by row 2 is "large".

The intuition behind (109) is as follows:

(110) a. if λ_2 is large, then update by row 2 will separate the W of M_1 from the L of M_2 by a large amount $\lambda_2 + 1$;

b. as that separation is large, it will take a while for it to be eroded by updates triggered by row 1;

c. thus, it will take a while for the current ranking vector to become again not OT-incompatible with row 2;

d. as row 2 can only trigger an update when not OT compatible with the current ranking vector, then row 2 won't have many chances of triggering an update.

As noted above, Boersma's update rule (10) and the cautious update rule (36) fit into the scheme (106) but differ for λ_2:

(111)
Boersma's update rule (10)	cautious update rule (36)
⏐	⏐
"small" $\lambda_2 = 1$	"large" $\lambda_2 = 2$

It turns out that the demotion amount $\lambda_2 = 1$ of Boersma's update rule (10) is too small for the algorithm to converge to the desired final ranking (104), as shown by counterexample (112) for $\theta^{\text{init}} = 7$ (but counterexamples can be constructed for every θ^{init}).

(112) $$\begin{bmatrix}0\\0\\7\\7\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}0\\1\\8\\6\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}1\\1\\7\\6\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}2\\1\\6\\6\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}3\\1\\5\\6\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}4\\1\\4\\6\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}4\\2\\5\\5\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}4\\3\\6\\4\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}5\\3\\5\\4\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}6\\3\\4\\4\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}6\\4\\5\\3\end{bmatrix}$$

It turns out instead that the demotion amount $\lambda_2 = 2$ of the cautious update rule (36) is large enough for the algorithm to converge to the desired final ranking (104), as proven in claim 5, using the invariant obtained in the preceding section. To illustrate, note that the algorithm with this promotion-demotion update rule and $\theta^{\text{init}} = 7$ admits only 13 learning paths, listed in (113): in all of them, row 2 triggers only two updates.

(113) $$\begin{bmatrix}0\\0\\7\\7\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}1\\0\\6\\7\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}2\\0\\5\\7\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}3\\0\\4\\7\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}4\\0\\3\\7\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}4\\1\\4\\5\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}4\\2\\5\\3\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}5\\2\\4\\3\end{bmatrix}$$
$$\xrightarrow{\mathbf{a}_2} \begin{bmatrix}3\\1\\5\\5\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}3\\2\\6\\3\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}4\\2\\5\\3\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}5\\2\\4\\3\end{bmatrix}$$
$$\xrightarrow{\mathbf{a}_1} \begin{bmatrix}4\\1\\4\\5\end{bmatrix} \xrightarrow{\mathbf{a}_1} \begin{bmatrix}5\\1\\3\\5\end{bmatrix} \xrightarrow{\mathbf{a}_2} \begin{bmatrix}5\\2\\4\\3\end{bmatrix}$$

$$
\begin{array}{ccccccccccccc}
 & & & & & & & & & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}4\\2\\5\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
 & & & & & & & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}2\\1\\6\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}3\\1\\5\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}4\\1\\4\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\1\\3\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
 & & & & & & & & & & & & & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}4\\2\\5\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
 & & & & & & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}3\\2\\6\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}4\\2\\5\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
 & & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}1\\1\\7\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}2\\1\\6\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}3\\1\\5\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}3\\2\\6\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}4\\2\\5\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
 & & & & & & & & & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}4\\1\\4\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\1\\3\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
 & & & & & & & & & & & & & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}4\\2\\5\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
\begin{bmatrix}0\\0\\7\\7\end{bmatrix} & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}0\\1\\8\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}1\\1\\7\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}2\\1\\6\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}3\\1\\5\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}4\\1\\4\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}4\\2\\5\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
 & & & & & & & & & & & & & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\1\\3\\5\end{bmatrix} & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix} \\
 & & & & & & & & & & & & & \xrightarrow{\mathbf{a}_2} & \begin{bmatrix}3\\2\\6\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}4\\2\\5\\3\end{bmatrix} & \xrightarrow{\mathbf{a}_1} & \begin{bmatrix}5\\2\\4\\3\end{bmatrix}
\end{array}
$$

The new cautious promotion-demotion update rule (36 was initially motivated from the point of view of *convergence*. The case of the comparative tableau (103) shows that this update rule might also be motivated from the point of view of *correctness*.

■ **Claim 4** The OT online algorithm run on input tableau (103) starting from the initial ranking vector (105) with the update rule (106) converges to the desired final ranking (104), *no matter how the two rows are sampled*, provided that total number α_2^{fin} of updates triggered by row 2 is small, namely:

$$\alpha_2^{\text{fin}} < \frac{\theta^{\text{init}}}{1+\lambda_2} \tag{114}$$

Proof. We want the final ranking vector $\boldsymbol{\theta}^{\text{fin}} = (\theta_{F_1}^{\text{fin}}, \theta_{F_2}^{\text{fin}}, \theta_{M_1}^{\text{fin}}, \theta_{M_2}^{\text{fin}})$ to represent the target ranking (104), repeated in (115).

$$\begin{array}{ccccccc} F_1 & \gg & M_1 & \gg & M_2 & \gg & F_2 \\ & | & & | & & | & \\ & \text{(a)} & & \text{(b)} & & \text{(c)} & \end{array} \tag{115}$$

Note that:

(116)
a. OT-compatibility with row 1 of (103) ensures that $\boldsymbol{\theta}^{\text{fin}}$ satisfies (115a);
b. assume that $\boldsymbol{\theta}^{\text{fin}}$ satisfies (115c);
c. then OT-compatibility with row 2 of (103) ensures that $\boldsymbol{\theta}^{\text{fin}}$ satisfies (115b).

Thus, it is sufficient to show that $\boldsymbol{\theta}^{\text{fin}}$ satisfies (115c), namely that:

$$\theta_{M_2}^{\text{fin}} > \theta_{F_2}^{\text{fin}} \tag{117}$$

Note that the final ranking values of M_2 and F_2 can be expressed as follows in terms of the total number α_2^{fin} of updates triggered by row 2:

$$\begin{array}{rcl} \theta_{F_2}^{\text{fin}} & = & \alpha_2^{\text{fin}} \\ \theta_{M_2}^{\text{fin}} & = & \theta^{\text{init}} - \lambda_2 \alpha_2^{\text{fin}} \end{array} \tag{118}$$

The inequality (117) thus becomes:

$$(1+\lambda_2)\alpha_2^t < \theta^{\text{init}} \tag{119}$$

which is the desired inequality (114). □

■ **Claim 5** The OT online algorithm run on input tableau (103) starting from the initial ranking vector (105) with the cautious update rule (36) satisfies (114) *no matter how the two rows are sampled*, and thus always converges to the desired final ranking (104).

Proof. Let me introduce the following notation:

$$\begin{array}{rcl} \alpha_1^t & = & \text{the number of times up to time } t \text{ that the algorithm has updated in the given run in response to row 1 of (103);} \\ \alpha_2^t & = & \text{the number of times up to time } t \text{ that the algorithm has updated in the given run in response to row 2 of (103).} \end{array} \tag{120}$$

The first step of the proof is to note that the two inequalities (121) hold; I will then derive from (121) the fact that (114) holds at any time t, and thus in particular at the final time.

(121)
a. $\alpha_1^t \leq 1 + \frac{1}{2}(\alpha_2^t + \theta^{\text{init}})$.
b. $\alpha_2^t \leq \min\left\{1 + \frac{1}{3}\theta^{\text{init}},\ 1 + \frac{1}{3}\alpha_1^t\right\}$.

Inequalities (121) are guaranteed by claim 3. For completeness, let me sketch the proof again. If $\alpha_1^t = 1$, then (121a) trivially holds. Otherwise, the proof of (121a) consists of the chain of implications in (122).

$$\begin{array}{rcl} \alpha_1^t \geq 1 & \Longrightarrow & \exists s < t \text{ s.t. } \alpha_1^s = \alpha_1^t - 1 \text{ and } \boldsymbol{\theta}^s \text{ not OT-compatible with } \mathbf{a}_1 \\ & \Longrightarrow & \exists s < t \text{ s.t. } \alpha_1^s = \alpha_1^t - 1 \text{ and } \theta_{F_1}^s \leq \theta_{M_1}^s \\ & \Longrightarrow & \exists s < t \text{ s.t. } \alpha_1^s = \alpha_1^t - 1 \text{ and } \alpha_1^s \leq -\alpha_1^s + \alpha_2^s + \theta^{\text{init}} \\ & \Longrightarrow & \exists s < t \text{ s.t. } \alpha_1^s = \alpha_1^t - 1 \text{ and } 2\alpha_1^s \leq \alpha_2^s + \theta^{\text{init}} \\ & \Longrightarrow & 2\Big(\alpha_1^t - 1\Big) \leq \alpha_2^t + \theta^{\text{init}} \\ & \Longrightarrow & \alpha_1^t \leq 1 + \frac{1}{2}(\alpha_2^t + \theta^{\text{init}}) \end{array} \tag{122}$$

Analogously, if $\alpha_2^t = 1$, then (121b) trivially holds. Otherwise, the proof of (121b) consists of (123) and (124).

$$
\begin{aligned}
(123)\quad \alpha_2^t \geq 1 \quad &\Longrightarrow \quad \exists s < t \text{ s.t. } \alpha_2^s = \alpha_2^t - 1 \text{ and } \boldsymbol{\theta}^s \text{ not OT-compatible with } \mathbf{a}_2 \\
&\Longrightarrow \quad \exists s < t \text{ s.t. } \alpha_2^s = \alpha_2^t - 1 \text{ and } \theta^s_{F_2} \leq \theta^s_{M_2} \\
&\Longrightarrow \quad \exists s < t \text{ s.t. } \alpha_2^s = \alpha_2^t - 1 \text{ and } \alpha_2^s \leq -2\alpha_2 + \theta^{\text{init}} \\
&\Longrightarrow \quad \exists s < t \text{ s.t. } \alpha_2^s = \alpha_2^t - 1 \text{ and } 3\alpha_2^s \leq \theta^{\text{init}} \\
&\Longrightarrow \quad 3\Big(\alpha_2^t - 1\Big) \leq \theta^{\text{init}} \\
&\Longrightarrow \quad \alpha_2^t \leq 1 + \frac{\theta^{\text{init}}}{3}
\end{aligned}
$$

$$
\begin{aligned}
(124)\quad \alpha_2^t \geq 1 \quad &\Longrightarrow \quad \exists s < t \text{ s.t. } \alpha_2^s = \alpha_2^t - 1 \text{ and } \boldsymbol{\theta}^s \text{ not OT-compatible with } \mathbf{a}_2 \\
&\Longrightarrow \quad \exists s < t \text{ s.t. } \alpha_2^s = \alpha_2^t - 1 \text{ and } \theta^s_{M_1} \leq \theta^s_{M_2} \\
&\Longrightarrow \quad \exists s < t \text{ s.t. } \alpha_2^s = \alpha_2^t - 1 \text{ and } -\alpha_1 + \alpha_2 + \theta^{\text{init}} \leq -2\alpha_2 + \theta^{\text{init}} \\
&\Longrightarrow \quad \exists s < t \text{ s.t. } \alpha_2^s = \alpha_2^t - 1 \text{ and } 3\alpha_2^s \leq \alpha_1^s \\
&\Longrightarrow \quad 3\Big(\alpha_2^t - 1\Big) \leq \alpha_1^t \\
&\Longrightarrow \quad \alpha_2^t \leq 1 + \frac{1}{3}\alpha_1^t
\end{aligned}
$$

In order to finally derive (114) from (121), let me distinguish two cases:

(125) a. $\alpha_1^t < \theta^{\text{init}} - 3$;
b. $\alpha_1^t \geq \theta^{\text{init}} - 3$.

In case (125a), (119) always holds, as shown by (126).

$$
(126)\quad \alpha_2^t \overset{(a)}{\leq} 1 + \frac{1}{3}\alpha_1^t \overset{(b)}{<} 1 + \frac{1}{3}\Big(\theta^{\text{init}} - 3\Big) = \frac{1}{3}\theta^{\text{init}}
$$

It is easy to see using (121b) that case (125b) can never arise. □

References

Bane, Max, and Jason Riggle. 2010. "The typological consequences of weighted constraints". In *Proceedings of CLS 45*.

Boersma, Paul. 1997. "How We Learn Variation, Optionality and Probability". In *IFA Proceedings 21*, 43–58. University of Amsterdam: Institute for Phonetic Sciences.

Cesa-Bianchi, Nicolò, and Gábor Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press.

Pater, Joe. 2008. "Gradual Learning and Convergence". *Linguistic Inquiry* 39.2:334–345.

Pater, Joe. 2009. "Weighted Constraints in Generative Linguistics". *Cognitive Science* 33:999–1035.

Prince, Alan, and Bruce Tesar. 2004. "Learning Phonotactic Distributions". In *Constraints in Phonological Acquisition*, ed. R. Kager, J. Pater, and W. Zonneveld, 245–291. Cambridge University Press.

Tesar, Bruce. 2007. "A Comparison of Lexicographic and Linear Numeric Optimization Using Violation Difference Ratios". Ms., Rutgers University; ROA-939.

Tesar, Bruce, and Paul Smolensky. 1998. "Learnability in Optimality Theory". *Linguistic Inquiry* 29:229–268.