Phonological time travel

Ollie Sayeed, Christ’s College, University of Cambridge

1 Outline

This paper is about two phonological effects involving ‘time travel’, where a process can be made to apply or not apply depending on what other processes are in its past or its future. First, I’ll illustrate Wilson’s (2006) ‘counterfeeding from the past’ using Optimality Theory with Candidate Chains (OT-CC); then, a new process of ‘counterfeeding from the future’, using both OT-CC and Classic OT. We’ll then see two analyses in Rule-Based Phonology (RBP) that compute the same grammars just by rule ordering; the question is then whether the presence or absence of these processes in natural languages give evidence for or against RBP or OT-CC, given that both of them are capable of computing processes extensionally equivalent to time travel. To begin answering this question, we look at the possible diachrony of time travel effects, and finally how learning algorithms in the two theories deal with time travel effects.

2 Counterfeeding from the past

2.1 OT-CC

OT-CC (McCarthy, 2007) extends Classic OT in two ways:

• changing the candidate output forms produced by Gen
• augmenting the set of constraints in Con.

Rather than evaluating candidates, we evaluate chains of candidates, more like derivations in RBP. McCarthy defines chains as subject to three conditions:

1. The first form in the chain has to be a faithful parse of the input.
2. Gradualness: every form in the chain incurs all the faithfulness violations of the previous form, plus exactly one more.
3. Local optimality: every form has to be more harmonic than the previous one according to the ranking of Con.

The second change from Classic OT is that as well as faithfulness and markedness constraints, Con contains precedence constraints evaluating properties of chains. Take two faithfulness constraints A and B; then the precedence constraint Prec(A,B) gets one violation mark for every one of two instances:

• a violation of A happens after a violation of B
• a violation of B happens without a violation of A before it.

McCarthy wants precedence constraints to be innate; but see Tihonova (2009) for the view that they can be acquired.

2.2 CFFTP

The fact that precedence constraints can judge properties of whole derivations, not just output forms, means OT-CC grammars can do ‘time travel’ - they’re allowed to depend on information from the derivation’s past or future. Wilson (2006) demonstrates ‘counterfeeding from the past’ (CFFTP) in OT-CC, where a process can blocked depending on what happened in its past, thanks to a particular ranking of precedence constraints.

Wilson’s example, (partly) based on Finnish (Kiparsky, 1973): suppose a language has three processes, expressed here in RBP.

- A: [+cons] → ∅ /...# (word-final deletion)
- B: e → i /...# (word-final raising)
- C: t → s /...i (assibilation)

In RBP, the possible kinds of ordering are the usual typology of feeding and counterfeeding. In OT-CC, we can add a twist: Wilson asks what would happen if we added a highly-ranked precedence constraint requiring that assibilation happens before deletion. Call this constraint Prec(C,A) for short, and have it ranked above the markedness constraint *ti that causes assibilation.

Given the input /tok/, the chain ⟨tok⟩ wins; so we have final deletion:

<table>
<thead>
<tr>
<th>/tok/</th>
<th>*C#</th>
<th>*e#</th>
<th>Prec(C,A)</th>
<th>*ti</th>
<th>Max</th>
<th>IO([HIGH])</th>
<th>IO([CONT])</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨tok⟩</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨tok, to⟩</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This incurs one violation of Prec(C,A), because there’s an instance of deletion without assimilation before it.

Thanks to Samuel Andersson and Bert Vaux for discussing these time travel effects with me; and to James Clackson, for his comments on the ancient Greek data.
Given /pek/, the chain ⟨pek, pe, pi⟩ wins, so we have deletion feeding raising:

<table>
<thead>
<tr>
<th>/pek/</th>
<th>*C#</th>
<th>*e#</th>
<th>Prec(C,A)</th>
<th>*ti</th>
<th>Max</th>
<th>IO(HIGH)</th>
<th>IO(CONT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pek)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(pek, pe)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(pek, pe, pi)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This also incurs one violation of Prec(C,A), for the same reason as before.

But from /tek/, the assibilation candidate ⟨tek, te, ti, si⟩ has two violations of Prec(C,A): one because there’s an instance of deletion without assibilation before it, and one because there’s an instance of assibilation after deletion. The winning chain ends up being ⟨tek, te, ti⟩, with no assibilation:

<table>
<thead>
<tr>
<th>/tek/</th>
<th>*C#</th>
<th>*e#</th>
<th>Prec(C,A)</th>
<th>*ti</th>
<th>Max</th>
<th>IO(HIGH)</th>
<th>IO(CONT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(tek)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tek, te)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tek, te, ti)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tek, te, ti, si)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

But assibilation is active in general in this grammar: from the input /te/, the assibilation candidate ⟨te, ti, si⟩ has no Prec(C,A) violations (because there are no instances of deletion-without-assibilation or assibilation-after-deletion), so it wins:

<table>
<thead>
<tr>
<th>/te/</th>
<th>*C#</th>
<th>*e#</th>
<th>Prec(C,A)</th>
<th>*ti</th>
<th>Max</th>
<th>IO(HIGH)</th>
<th>IO(CONT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(te)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(te, ti)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(te, ti, si)</td>
<td>-</td>
<td>!*</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- In other words, assibilation is blocked in this grammar exactly when fed by deletion earlier in the derivation.
- Whether or not a process applies at some point in a chain can depend on the earlier parts of the chain, and not just the immediately preceding form.

Wilson points out that CFFTP can be ‘stretched’ arbitrarily far, with processes being able to look back arbitrarily many steps in the derivation: if deletion itself were fed by some process violating faithfulness constraint Z, then the precedence constraint Prec(C,Z) would have the same effect of blocking violations of C in exactly those chains that earlier involved violations of Z.

3 Counterfeeding from the future

3.1 CFFTF in OT-CC

I want to raise a different kind of ‘time travel’, in some sense a mirror image - whether or not a process applies can be affected by the future of the derivation. To emphasize the parallel with Wilson’s effect, I call this ‘counterfeeding from the future’ (CFFTF).

- Suppose we have the same three processes, A (deletion), B (raising), and C (assibilation) as in the pseudo-Finnish example for CFFTP.
- This time, try ranking the markedness constraint *e# (causing raising) lower than both Prec(C,A) and *ti.
- Prec(C,A) is now even higher-ranked than before, outranking everything except *C#.

Given the input /tok/, as in the CFFTP case, the chain (tok, to) is the winner; so the CFFTF grammar has deletion on its own:

<table>
<thead>
<tr>
<th>/tok/</th>
<th>*C#</th>
<th>Prec(C,A)</th>
<th>*ti</th>
<th>*e#</th>
<th>Max</th>
<th>IO(HIGH)</th>
<th>IO(CONT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(tok)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tok, to)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Given an input /pek/, the chain ⟨pek, pe, pi⟩ wins, as before:

<table>
<thead>
<tr>
<th>/pek/</th>
<th>*C#</th>
<th>Prec(C,A)</th>
<th>*ti</th>
<th>*e#</th>
<th>Max</th>
<th>IO(HIGH)</th>
<th>IO(CONT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pek)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(pek, pe)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(pek, pe, pi)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

But for the input /tek/, which was the time-travelling input in the CFFTP case, we see a different type of effect with the higher ranking Prec(C,A). The assibilation candidate ⟨tek, te, ti, si⟩ still loses, for incurring two violations of Prec(C,A):

<table>
<thead>
<tr>
<th>/tek/</th>
<th>*C#</th>
<th>Prec(C,A)</th>
<th>*ti</th>
<th>*e#</th>
<th>Max</th>
<th>IO(HIGH)</th>
<th>IO(CONT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(tek)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tek, te)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tek, te, ti)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tek, te, ti, si)</td>
<td>-</td>
<td></td>
<td>-</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Not only does (tek, te, ti, si) lose, the raising candidate (tek, te, ti) also loses, now that *ti is higher-ranked than *e#. The winner is (tek, te) - we only see deletion in this CFFTFT grammar, and not raising or assimilation.

Like CFFTP, this process can be stretched:

- If C in turn feeds some process D, then the precedence constraint Prec(D,A) could be highly ranked, penalizing the candidate violating D.
- All the markedness constraints except the one associated with B can then be the next highly ranked, meaning the winning candidate is the chain stopping just short of violating B.
- The interpretation to draw is the mirror image of CFFTP: in this CFFTFT grammar, raising is blocked in exactly those chains in which it would lead to assimilation in its future.

3.2 CFFTTF in Classic OT

- The ‘time travel’ feature of CFFTFT - that it can look arbitrarily far into the future of the derivation - is actually also a feature of Classic OT, and not dependent on precedence constraints (unlike CFFTP, as far as I can see).
- For this CFFTFT example to work, all we need is that some constraint is violated by the assimilation candidate in those cases where it was fed by raising.

Just by looking at the tableau for /tek/ under the OT-CC grammar, we see that we could achieve the same effect by ranking IO(CONT) higher in the hierarchy:

<table>
<thead>
<tr>
<th>/tek/</th>
<th>*C#</th>
<th>*ti</th>
<th>IO(CONT)</th>
<th>*e#</th>
<th>MAX</th>
<th>IO(HIGH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[tek]</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[te]</td>
<td></td>
<td></td>
<td></td>
<td>*!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ti]</td>
<td></td>
<td></td>
<td></td>
<td>*!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[si]</td>
<td></td>
<td></td>
<td></td>
<td>*!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IO(CONT) can’t be ranked higher than *ti, or else the grammar just wouldn’t contain assimilation as a process at all. But with IO(CONT) ranked just below *ti and higher than *e#, the grammar avoids both raising and assimilation to stick with [te]. When raising wouldn’t feed assimilation, as in the input /pek/, we do see raising:

<table>
<thead>
<tr>
<th>/pek/</th>
<th>*C#</th>
<th>*ti</th>
<th>IO(CONT)</th>
<th>*e#</th>
<th>MAX</th>
<th>IO(HIGH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[pek]</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[pe]</td>
<td></td>
<td></td>
<td></td>
<td>*!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[pi]</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This kind of Classic OT grammar, then, has the CFFTTF property: raising is blocked just when it would lead to assimilation in its future.

- The Classic OT implementation of CFFTTF can also be stretched arbitrarily far into the future, like the OT-CC implementations of both CFFTP and CFFTFT.
- If assimilation feeds some process Z, the faithfulness constraint for Z can be ranked to just below the markedness constraint for Z, just as the precedence constraint linking Z and A is highly ranked in the CFFTFT grammar in OT-CC.

N.B. OT processes are obviously sensitive to the ‘far future’, in that all markedness constraints refer to the surface, but this CFFTTF is more general: the potential future process causing blocking of an earlier one can be arbitrarily deep below the surface level, so a process can be sensitive to any point in its future and not just the very end. Another difference in interpretation is that although OT grammars can commonly involve processes being blocked by particular undesirable surface configurations, what’s specifically blocking the first process here is the potential application of another process, rather than a static configuration.

4 Some ‘juggling’ rule-based grammars

- Wilson: ‘If counterfeeding from the past were found in natural languages, this would constitute a strong empirical argument in favor of OT-CC.’
- In the case of CFFTTF, too, the fact that markedness constraints can be highly ranked to block possible later stages of the derivation seems not to have an analogue in an RBP grammar involving ordered rules.

But both CFFTP and CFFTTF can be computed by an RBP grammar, using a trick I’ll call ‘juggling’.

- The machinery of a juggler’s two hands isn’t enough to manipulate three balls in separate ways at once, but by throwing one of the balls up in the air, the juggler can separately manipulate the two other balls before the third ball falls back down again.
- By analogy, suppose we ‘throw’ the problematic segments out of the picture by mapping them to some segment not affected by later rules, and bring them back in at the end of the derivation.
• In the case of CFFTP, we want instances of /i/ ultimately fed by deletion of a final consonant not to cause assibilation of a preceding /t/. This is possible, if we make sure that derivations starting with deletion don’t have an /i/ in them at the time that assibilation applies; we map the offending segment to something else.

Taking instances of /e/ that would ultimately cause assibilation and mapping them to some other segment /X/, we get the following grammar:

1. \(e \rightarrow X / t \_ C\#\)
2. \(C \rightarrow \emptyset / \_ \#\)
3. \(e \rightarrow i / \_ \#\)
4. \(t \rightarrow s / \_ i\)
5. \(X \rightarrow i\)

/X/ is mapped to /i/ at the end of the derivation. This generates the right input-output mappings:

• Given /pek/, we have /pek/ \(\rightarrow [pXk] \) (by 1) \(\rightarrow [pX] \) (by 2) \(\rightarrow [pi] \) (by 5), so the illusion is that both deletion and raising apply as expected.
• Given /te/, we have /te/ \(\rightarrow [ti] \) (by 3) \(\rightarrow [si] \) (by 4), so raising and assibilation both apply; rule 1 only applies to /e/ before a word-final consonant, so it doesn’t apply here.
• But given /tek/, we have /tek/ \(\rightarrow [tXk] \) (by 1) \(\rightarrow [tX] \) (by 2) \(\rightarrow [ti] \) (by 5), with no application of 3 or 4. By making sure that there were no instances of word-final /e/ fed by deletion, we’ve made sure that there were no instances of assibilation fed by deletion, and so this RBP grammar has the CFFTP property: assibilation is blocked in exactly those cases when it would have been fed by deletion.

Surprisingly, an RBP grammar can compute CFFTF by almost an identical set of rules. In the CFFTP grammar, in derivations that began with deletion potentially feeding assibilation, we wanted deletion and raising to happen but no assibilation: deletion was done by a rule, and raising was done implicitly in a ‘juggling’ way by mapping /e/ to /X/ and then back to /i/. CFFTF is very similar to state: in derivations that potentially end in assibilation, we want deletion to happen but no raising or assibilation. In other words, we want the same ruleset, but without the juggling rule implementing raising.

• We do this by mapping /X/ back to the unraised /e/, rather than /i/.
• We also want to target deletion cases that would later feed assibilation - not just any deletion cases - so the juggling rule 1 will need to apply to final consonants following a /t/ before the vowel.

This is the RBP grammar that generates CFFTF:

1. \(e \rightarrow X / t \_ C\#\)
2. \(C \rightarrow \emptyset / \_ \#\)
3. \(e \rightarrow i / \_ \#\)
4. \(t \rightarrow s / \_ i\)
5. \(X \rightarrow e\)

Like in the CFFTP case, this grammar gives the right input-output mappings for CFFTF:

• Given /pek/, we have /pek/ \(\rightarrow [pe] \) (by 2) \(\rightarrow [pi] \) (by 3); because 1 specifies the /e/ needs to be preceded by a /t/, it doesn’t apply.
• Given /te/, we have /te/ \(\rightarrow [ti] \) (by 3) \(\rightarrow [si] \) (by 4); 1 also doesn’t apply here, because there’s a following consonant.
• Given /tek/, we have /tek/ \(\rightarrow [tXk] \) (by 1) \(\rightarrow [tX] \) (by 2) \(\rightarrow [te] \) (by 5), with no application of 3 or 4 as in the CFFTP case. It’s only in the specific t _ C# environment that the juggling rule applies - essentially ‘freezing’ the /e/ to the effects of any further rules - so we have the right interpretation for CFFTF: raising fails to apply, in exactly those derivations that would later lead to assibilation.

So RBP can compute both CFFTP and CFFTF, with the help of a juggling rule. The /X/ juggled to is an unknown, rather than an abstract, segment - it could be /e/ or /i/ or /j/, or anything else at all not otherwise present in the language.

This strategy is clearly a kludge /ˈklʌʤ/; it ‘gets’ time travel effects - or rather, something extensionally equivalent to time travel effects - but in an ugly way using a ‘Duke of York’ derivation. But it can manage it. After all, what we can see by observing language is the input-output mapping, the grammar in extension; if we do find time travel effects in extension, there’s no guarantee they need to be computed the OT-CC way in intension.
• If time travel effects appear in natural languages (see Wolf, 2010 for arguments that CFFTP does), then undergeneration would be a serious problem for RBP, but we’ve shown that RBP doesn’t undergenerate. Handed a time-travel-type input, an RBP grammar is capable of fitting it - however messy.

• If time travel effects don’t appear, however, there isn’t necessarily a problem for OT-CC either: not every grammar computable by the phonological component has to exist, as there might be other reasons (say, diachronic ones) why some computable grammar isn’t attested.

So RBP can deal with time travel, and OT-CC can deal with the absence of time travel. The question now is: does the presence or absence of time travel from the natural languages of the world give us any evidence to judge between RBP and OT-CC?

5 Diachrony of time travel processes

5.1 Possible sources of a phonological process

Intuitively, OT-CC gets time travel effects more ‘naturally’ than RBP; but this is meaningless without an explanation of why this means time travel should be more common under OT-CC than under RBP. We need to know how a time travel process might get into the grammar - in other words, what would lead to time travel being acquired.

In general, using terms from Jesney (2005), a phonological process might be:

- target-like - an accurate copy of a process present in the input
- non-target-like - ‘mislearnt’, by a misinterpretation of the input (or by quirks of the learning algorithm with nothing to do with the input at all).

So to know how easy it is for the learning algorithm of a particular theory to learn a time travel effect, we need to know:

- how easy it is to accurately learn a time travel effect from the input
- how easy it is to mislearn a non-time-travel input as involving time travel.

We’ll address these questions in the next section. Before considering acquisition directly, though, a prerequisite question is how a time travel effect - or a process in the synchronic phonology in general - might appear in the input in the first place.

• One option is that the process is the phonologization (Bermudéz-Otero, 2014) of an originally physical, phonetic effect. Final devoicing is of this type: for a handful of articulatory and acoustic reasons, voiced obstruents at word ends are mispronounced or misperceived as voiceless, and this is interpreted by speakers as the output of a phonological rule.

• Alternatively, it might be the consequence of a series of rules, compressed by learners into a single rule that (while an accurate input-output mapping) wasn’t in the grammars of the generation before; the most simple way this happens is by ‘telescoping’ (Kenstowicz and Kisseberth, 1977, chapter 2). For example, the Ukrainian rule $e \rightarrow i$ / - C # (described by K&K) is the telescoped outcome of a series of sound changes $e > e$: $> i$: $> i$.

In the case of time travel effects, I rule out the first option, because (I take it) it isn’t possible for a purely physical, phonetic effect outside speakers’ control to have a time-travel-type structure. It isn’t meaningful for a physical process of consonant deletion to ‘feed’ a physical process of assibilation of a stop earlier in the syllable - that would involve real time travel!

This leaves the second option: if there are time travel effects in the input, they have to come from telescoping of a series of rules, which the child analyses as a single time-travelling process. The next relevant question is what series of historical changes could be telescoped into a synchronic time travel effect.

5.2 A non-example: clusters and *â in Ionic Greek

A claimed example of CFFTP in a synchronic grammar comes from Adams’ (1972) interpretation of Kiparsky’s (1973) history of sonorant clusters in Ionic Greek.

Adams’ claim is that the behaviour of long /â/ in the synchronic Ionic grammar depends on what rule earlier in the derivation created it: these earlier rules are synchronic reflexes of historical compensatory lengthening sound changes caused by simplification of certain clusters in earlier Greek.

Adams points out that historically, where long *â was created by compensatory lengthening after deletion of an *s in a sonorant cluster, it raises to é:

- éphâna aor. ‘I showed’ < *éphâna < *éphan-sa
- selâna ‘moon’ < *selânâ < *selánsnâ

But when *â was created by compensatory lengthening in a cluster with *w, it stays in Ionic as â:
• kalós ‘good’ < *kalwós
• pththánš ‘I arrive’ < *pθthán-wɔ

If the Ionic Greek grammar contains a synchronic process /ā/ → /e/, it’d need to have a CFFTP flavour: ‘turn /ā/ → /e/, except when fed by compensatory lengthening from a /w/ cluster earlier in the derivation’.

This process, as we said earlier, has to come from a telescoping of a series of sound changes - and we can see their chronology in their distribution around the dialects.

• The lengthening from *s clusters (the ‘first compensatory lengthening’) is also shared with the Attic and Doric dialects - to the exclusion of Aeolic, which simplifies these clusters by geminating the consonant instead.
• The fronting of *ā created a vowel that was originally *d; Cycladic inscriptions spell the vowel as <H>, distinct from both *ē <E> and *a <A> (Sihler, 1995). This fronting is also shared with Attic, but not Doric or Aeolic.
• The lengthening from *w (the ‘third compensatory lengthening’; the second involves *ns- clusters of various sources) is a special development of Ionic - in Attic, Doric, and Aeolic, we have loss of *w leaving a short vowel in kalós (Miller, 2013).

So within Ionic, we have a clear relative chronology:

First CL ⇒ fronting of *ā ⇒ third CL

Adams’ assumption\(^2\) is that speakers will collapse the two compensatory lengthening sound changes into one synchronic rule, even though they happened separately in history - and this collapsing into a single lengthening rule is the telescoping that produces the claimed CFFTP effect in synchronic Ionic.

I’ve included this example to illustrate the spirit of a diachronic explanation of a time travel effect: it needs to involve the telescoping of multiple sound changes into one synchronic rule. I think this isn’t good enough evidence for a CFFTP effect in Greek, for a couple of reasons:

• The rounds of compensatory lengthening are expressed by different rules - *-wR- behaves differently to *-Rw-, for example, and the *-sR- and *-Rs- lengthenings were separate changes happening in different dialects (Clackson, p.c.) - so it’s not clear speakers would need to analyse them as a single process, which is key to the CFFTP interpretation.

More importantly, it’s also not clear there’s enough evidence in the input to set up both the compensatory lengthening changes as synchronic rules. In the case of the first CL, speakers would have evidence from the signmatic aorist marked by -s-, as in épʰênã < *éphan-sa (with stem phan-). But examples of the third CL in paradigms aren’t in any way common: pththánš < *pθthán-wɔ is one of a small handful coming from forms with a *nu-suffix that were thematized with a following vowel early in Greek, causing the *u to be syllabified as a glide *w (Sihler, 1995).

Of course, there are more examples of CFFTP than this in the literature (Wolf, 2010): if other examples have diachronic explanations, they’ll have to be analysed in this spirit, finding a series of telescappable sound changes. To tell how likely these series of sound changes are, we need more diachronic analyses of other, more reliable synchronic time travel effects.

An interesting point: if CFFTP comes from telescoping, then it’s pure chance that OT-CC happens to have the machinery to express it in a neat way - because the way OT-CC expresses it doesn’t reflect the way the process actually arose (which, diachronically, looks more like the ugly RBP grammar!).

### 6 Is time travel (mis)learnable?

The jury’s out, then, as to whether time travel effects can easily be created by telescoping sound changes. Now to the acquisition question: how easy is it either to learn target-like time travel, or to innovate non-target-like time travel, in RBP or OT-CC?

We can summarize what would be at stake, for each theory:

• If time travel effects are common diachronically, and easy to learn accurately, then we predict they should be common in natural languages.
• If time travel effects easily appear spontaneously through mislearning of the input, then we also predict they should be common in natural languages.
• If time travel effects are both hard to learn accurately and hard to innovate, we predict they should be rare in natural languages.

### 6.1 RBP learning

In RBP, learning involves postulating an ordered series of rules: the question for an RBP learning algorithm is when exactly rules should be postulated. In particular, it needs to be able to acquire ordering, as well as the individual rules.
One simple algorithm for learning rules (Vaux and Nevins, 2008) learns them
one-by-one: two rules are learnt independently - each under conditions where
only one of them applies.

If the two rules are potentially in a feeding or bleeding relationship, the child
needs to look for evidence as to which way they’re ordered: this comes by
looking at the case where they both apply.

This one-by-one learning algorithm could never acquire either target-like or
non-target-like time travel: each rule has to be learnt from independent evidence,
and there can’t be any evidence (outside the time travel derivations) of the juggling
rules in our RBP accounts of time travel effects.

A more sophisticated RBP learning algorithm, though, might be able to learn
multiple rules in one go: it could ‘take stock’ of the whole phonology, before
proposing a set of ordered rules that accounts for the data. (Or it might arrive at
this by some process, like algorithmically modifying grammars until they fit the
input.)

This algorithm, by definition, can learn any RBP grammar - the question is whether
time travel effects are easy or hard for it. Here’s a table of what this algorithm
would have to ‘notice’ in order to acquire various kinds of rule interaction (where
A is the rule ordered before B):

<table>
<thead>
<tr>
<th>Rule interaction</th>
<th>Environments the learner needs to notice</th>
</tr>
</thead>
<tbody>
<tr>
<td>A feeding B</td>
<td>A on its own, B on its own, A and B together</td>
</tr>
<tr>
<td>A bleeding B</td>
<td>A on its own, B on its own, A but not B</td>
</tr>
<tr>
<td>B counterfeeding A</td>
<td>A on its own, B on its own, B but not A</td>
</tr>
<tr>
<td>B counterbleeding A</td>
<td>A on its own, B on its own, A and B together</td>
</tr>
<tr>
<td>A CFFTPing B</td>
<td>A on its own, B on its own, A but not B</td>
</tr>
<tr>
<td>B CFFTPing A</td>
<td>A on its own, B on its own, neither A nor B</td>
</tr>
</tbody>
</table>

To learn CFFTP, the learner has to notice A, B, and that B is blocked when
fed by A; this isn’t any more complicated to learn than bleeding, which also
requires noticing that A and B happen independently but that B is blocked by
the application of A.

CFFTF is arguably harder: it involves noticing that A and B apply separately,
but that A doesn’t apply when it would feed B. This is harder, in that the
crucial fact being spotted is the absence of an event; neither A nor B actually
apply to the crucial forms!

In this model, too, non-target-like time travel is hard to acquire. Intuitively, an
algorithm that runs through possible grammars would be unlikely to propose one
as complicated as the grammars needed to compute time travel effects; but this
needs a more precise statement of the learning algorithm to say any more.

### 6.2 OT-CC learning

In learning an OT-CC grammar, the set of constraints Con is already innate:
what needs to be learnt is the ranking of Con. The statements of what rankings
constitute time travel effects are simple:

- **CFFTP** happens when \( \text{Prec}(C,A) \gg C \), where C is a markedness constraint,
  and markedness constraints for intervening processes outrank \( \text{Prec}(C,A) \).
- **CFFTF** happens when \( \text{Prec}(C,A) \gg C \), where C is a markedness con-
  straint, and markedness constraints for intervening processes are outranked
  by \( \text{Prec}(C,A) \) and C.

An early algorithm for acquisition of any kind of OT was **Recursive Constraint
Demotion** (Tesar and Smolensky, 2003; RCD), which learns a grammar by starting
with the innate hierarchy - where all markedness constraints are ranked above all
faithfulness constraints - and responds to input-output pairs by recursively reranking
the constraints until settling on a grammar that accounts for the data. Specifically:

- Start with a normal OT tableau, evaluating the input according to the innate
  ranking of Con.
- For each losing candidate and each constraint, subtract the number of viola-
  tions from the number of violations of the winning candidate. This gives a
  value of either W (‘favours the winner’) or L (‘favours the loser’) in each cell.
- Demote all the constraints that favour some loser, i.e. that have some L in
  their column. This creates a new stratum of constraints, ranked below the
  rest.
- Take in a new input-output pair, and repeat.

To get a time travel effect, we need the precedence constraint \( \text{Prec}(C,A) \) to
end up outranking the corresponding markedness constraint C. Tihonova (2009)
assumes that precedence constraints start out in the middle of the hierarchy,
outranking all faithfulness constraints and outranked by all markedness constraints.

For \( \text{Prec}(C,A) \) to come to outrank C, then, C will have to be demoted to some
position below \( \text{Prec}(C,A) \) - meaning that C will have to favour some loser in a
situation where \( \text{Prec}(C,A) \) favours the winner. This will be true in derivations
where the process C happens but A doesn’t; using our earlier example, where
assibilation happens without being fed by deletion.

If these scenarios where C happens without A are enough to cause C to be demoted below Prec(C,A), we have a nascent time travel effect. The ranking of other markedness constraints relative to Prec(C,A) will determine whether this effect is a kind of CFFTP or a kind of CFFTF.

- Of course, there’s another way to reach CFFTF in OT-CC, which is the way Classic OT does it: by ranking a faithfulness constraint F above the markedness constraint M for a process that would feed the process violating F. Given the existence of CFFTP, we reject Classic OT for the cardinal sin of undergenerating, but OT-CC can still use the same method by ignoring its precedence constraints.

Both of these ways of getting time travel effects might be predicted to be common, then, as stages in the acquisition process (see Tihonova, 2009 for an OT-CC account of children’s spontaneous opacity). As ways of getting target-like time travel, they’re well capable, like the more powerful RBP algorithm.

The problem with judging predictions about non-target-like time travel, though, is this algorithm applies recursively given new inputs: and sooner or later, if the target grammar doesn’t contain time travel effects, they ought to be unlearnt. To tell whether or not OT-CC predicts that non-target-like time travel should be spontaneously innovated, we need a more precise model of when exactly the learning algorithm decides to ignore all future data and stick with the current hierarchy.

7 Conclusion

In sum - we’ve seen two kinds of effects that can be grouped under the label of ‘time travel’, in which some process is sensitive to the past or future of the derivation rather than just its immediately preceding state.

- CFFTP, modellable in OT-CC, is where a rule can be blocked by some other rule applying in its past
- CFFTF, modellable in both OT-CC and Classic OT, is where a rule can be blocked by some other rule that would have applied in its future.

Both of these effects are also computable in RBP, using grammars that are intuitively messy but necessary to capture the data.

The question then is how likely time travel effects are to appear in RBP vs OT-CC grammars, rather than just whether they’re computable. We’ve seen that this question has two parts: i) how do time travel effects appear diachronically? and ii) how easily can they be (mis)learnt from the input? These questions are both inconclusive; we need further analyses of the diachrony of attested time travel effects, and more precise formulations of learning algorithms to tell how likely non-target-like time travel should be. Both the more powerful RBP algorithm and the RCD algorithm for OT, though, are capable of learning target-like time travel effects.

8 References