

Efficient Evaluation and Learning in Multilevel Parallel Constraint Grammars

Paul Boersma

Jan-Willem van Leussen

In multilevel parallel Optimality Theory grammars, the number of candidates (possible paths from the input to the output level) increases exponentially with the number of levels of representation. The problem with this is that with the customary strategy of listing all candidates in a tableau, the computation time for evaluation (i.e., choosing the winning candidate) and learning (i.e., reranking the constraints on the basis of language data) increases exponentially with the number of levels as well. This article proposes instead to collect the candidates in a graph in which the number of nodes and the number of connections increase only linearly with the number of levels of representation. As a result, there exist procedures for evaluation and learning that increase only linearly with the number of levels. These efficient procedures help to make multilevel parallel constraint grammars more feasible as models of human language processing. We illustrate visualization, evaluation, and learning with a toy grammar for a traditional case that has already previously been analyzed in terms of parallel evaluation, namely, French liaison.

Keywords: Optimality Theory, learning algorithm(s), multilevel models, French, phonology, computational linguistics

1 Multilevel Parallel Constraint Grammars

Difficult problems in phonology and its interaction with phonetics, such as the French liaison problem discussed in this article, often benefit from being expressed in terms of multiple levels of representation. The five levels considered here are the ones shown in figure 1: *meaning*, *morphemes*, (phonological) *underlying form*, (phonological) *surface form*, and *phonetic form*. In an Optimality Theory (OT) implementation (Apoussidou 2007, Boersma 2007), relations between adjacent levels are evaluated by interlevel constraints (here, *lexical-semantic*, *lexical-phonological*, *faithfulness*, and *cue* constraints), and the representations themselves are evaluated by intra-level constraints (here, *morphosyntactic*, *structural*, and *articulatory* constraints). The downward

We acknowledge the support of grant 277.70.008 of the Netherlands Organization for Scientific Research (NWO), the input from audiences at a lecture series in Tromsø (November 2007), the 18th Manchester Phonology Meeting (May 2011), the Workshop on the Status and Use of Corpora in Linguistics in Montpellier (June 2012), and a workshop about the PFC (Phonologie du français contemporain) Corpus in Toulouse (November 2012). We thank Tamás Biró for his extensive comments on an earlier version of this article.

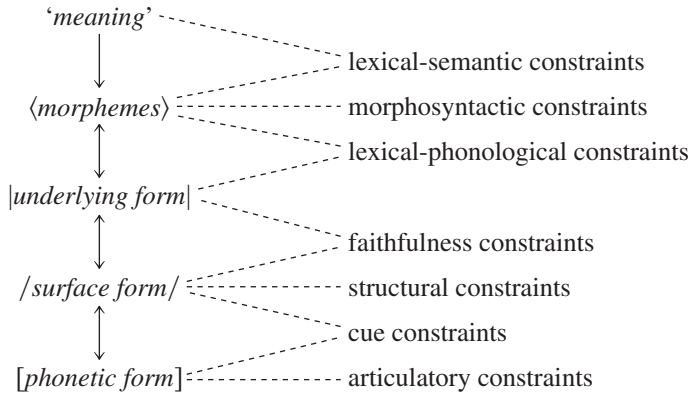


Figure 1

The five-level grammar model needed for this article

arrows in the figure represent the direction of the production process; the input to this process is an intended meaning and its output is a realized phonetic form.

For this article, the *parallel* property of the model is crucial: production is implemented as the evaluation of candidates across all levels of processing in parallel. The speaker starts with an intended meaning and computes an optimal quadruplet of morphemes, underlying, surface, and phonetic form. Cross-level parallelism means that (e.g.) faithfulness constraints, which evaluate the relation between underlying and surface form, can interact with cue constraints, which evaluate the relation between (phonological) surface form and (auditory-)phonetic form. This cross-level parallelism allows “later” phonetic considerations such as articulatory effort and auditory cue quality to influence “earlier” phonological decisions (Boersma 2007, 2008). In this article, parallelism crucially allows “later” phonological considerations such as hiatus avoidance to influence “earlier” choices in the morphology such as gender selection. These types of bottom-up influences in production are depicted with upward arrows in figure 1.

In *learning*, cross-level parallelism typically causes the credit (or blame) for overt phenomena to be distributed over multiple levels. Thus, when an underlying $[an+pa]$ is pronounced as phonetic $[ampa]$, the learner will automatically come to interpret this partly as a phonological and partly as a phonetic assimilation (Boersma 2008).

A potential problem with parallel evaluation is its computational load. If every meaning comes with 10 possible morphemes, 10 possible underlying forms, 10 possible surface forms, and 10 possible phonetic forms, and we assume that all of these can be combined freely, then the number of candidates (quadruplets of morphemes, underlying, surface, and phonetic form) that have to be evaluated for any given meaning is 10,000. In other words, the number of candidates increases exponentially with the number of levels of representation. If we list all of these candidates in a big tableau, and use the usual candidate elimination procedure for tableaux (Prince and Smolensky 1993), the computational load of choosing the optimal candidate increases exponen-

tially with the number of levels of representation. This problem could become especially prohibitive in a practical comprehensive computational model of linguistic processing, in which there may be more phonological levels than in figure 1 (e.g., an additional word level; Mohanan 1981, Kiparsky 1982, Bermúdez-Otero 2003) and more phonetic levels (e.g., separate articulatory and auditory levels; Boersma 1998), and in which there will certainly be more semantic levels (e.g., separate lexical and sentence levels; Jackendoff 1997) as well as multiple syntactic levels (e.g., deep and surface structure; Chomsky 1957) and some discourse levels (Hengeveld and Mackenzie 2008); a total of 12 or more levels can easily be imagined.

Fortunately, an exponentially increasing candidate set does not have to come with an exponentially increasing computation time. The computational OT literature has repeatedly shown that a candidate set whose size is exponential in a certain parameter can be evaluated in a time that is linear in that parameter, if certain conditions are met. For instance, in two-level OT, the number of output candidates is typically exponential in the length of the input: if (e.g.) each input segment either surfaces unchanged, or is deleted, or has something epenthesized before and/or after it, each input segment has four potential surface realizations, and an input string of N segments has 4^N surface candidates. However, Ellison (1994) showed that if such a candidate set can be represented as a regular expression and appropriate restrictions on the constraints are met (“finite-state OT”), the evaluation time is only linear in the length of the input (Riggle (2004) improves on this, by showing that in finite-state OT even an infinite candidate set—for example, one with unbounded epenthesis—can be evaluated in linear time). Likewise, if we allow subsegmental phonological structure, the number of candidates is exponential in the number of autosegmental tiers, but Eisner (1997) showed that if in finite-state OT the candidate set can be represented on a nonhierarchical set of tiers with a timeline (“Primitive OT”), the evaluation is linear in the number of tiers.

The same consideration applies to the multilevel OT of figure 1, where the number of candidates is exponential in the number of levels of representation. In this article, we show that if each constraint evaluates either a single level or the relation between two adjacent levels, evaluation time becomes linear in the number of levels. As in Ellison 1994, Eisner 1997, and Riggle 2004, this linearity is achieved with graph-theoretic methods, but without requiring the assumptions of finite-state OT. Specifically, we represent the candidate set not in a big tableau but in a *candidate graph*, which would reduce the 10,000-candidate case above to a graph with only 40 nodes and 310 connections—that is, with a number of elements that increases only linearly with the number of levels of representation.

The concept of the candidate graph is introduced in section 2. In section 3, we then show how this economical representation naturally leads to efficient procedures for evaluation and learning, which either extend the usual candidate elimination procedure for tableaux (Prince and Smolensky 1993) or reflect standard procedures for optimization in graphs (Ford 1956, Bellman 1957, Viterbi 1967; for OT, Riggle 2009). As a real-language case to illustrate the procedures of section 3, in section 2 we introduce the interaction of liaison and gender selection in French. In section 4, we investigate how the learning procedure works out for the French case. In section 5, we discuss how our findings relate to complexity-reducing proposals for other parts of OT.

2 Visualization of Candidates with Multilevel OT Tableaux

The usual way to visualize an evaluation process in OT is with a *tableau*, a list of all possible candidate outcomes with their constraint violations (Prince and Smolensky 1993). For parallel models such as the one in figure 1, this list can become very long. The present section illustrates this point with the example used throughout this article: the phenomenon of phonological *liaison* in French and its interaction with morphological gender selection. Liaison has served as a testing ground for many phonological theories (for reviews, see Eychenne 2006, Durand and Lyche 2008); its parallel interaction with gender selection has been noted by Encrevé-Lambert (1971) and Encrevé (1988) and was first addressed within OT by Tranel (1996). We first present a traditional serial analysis, then proceed to the parallel analysis.


2.1 The Serial Analysis of Gender Selection and Liaison

The serial analysis of gender selection and liaison in French proceeds as follows, in a stepwise OT evaluation where in each step we heavily restrict the candidate generator (GEN) to providing only the most relevant candidates.

We only model the behavior of the adjective ‘good’ in French, which is pronounced [bɔ̃n] when feminine and [bɔ̃] when masculine, except that before vowel-initial nouns it is always [bɔ̃n] (in first approximation). For our serial account, we follow the early generative approach by Schane (1968), Dell (1970, 1973), and Selkirk (1972), which posits that in the underlying form there is a single stem [bɔ̃n] for both masculine and feminine gender, and that a gender morpheme is appended to this, which is phonologically empty ($|\emptyset|$) for the masculine and a schwa ($|\ə|$) for the feminine.

Suppose that one wants to produce the meaning ‘good_i actor_i’ in French. In a serial version of figure 1, a French speaker starts by connecting this meaning to the French morphemes $\langle \text{bon-M}; \text{acteur}_M \rangle$, where the subscript *M* marks the [masculine] value of the gender feature of the noun *acteur*, and the appended *-M* indicates the masculine ending of French adjectives. In OT, this could look like the following tableau:

(1) Mapping meaning to morphemes


‘good _i actor _i ’	*⟨FM⟩
 ⟨bon-M; acteur _M ⟩	
⟨bon-F; acteur _M ⟩	*!

Here, the constraint *⟨FM⟩ militates against a gender mismatch within the morphemic level. In the serial account, the working of this constraint is quite trivial: since it interacts with nothing else, it will always force a gender identity between adjective and noun, as it does here.

Next, the serial speaker connects the morpheme sequence to an underlying form by using a ranking of lexical-phonological constraints (Boersma 2001, Apoussidou 2007) that makes sure that the underlying forms proposed by Schane, Dell, and Selkirk are selected. For the present

case, the result is $|b\text{ɔ}n+\emptyset\#\text{akt}\text{ɛ}\text{ɛ}|$ in (2), where “+” denotes a word-internal morpheme boundary and “#” a word boundary.


(2) *Mapping morphemes to an underlying form*

$\langle bon-M; \text{acteur}_M \rangle$	* $\langle bon-M \rangle$ bɔ̃	* $\langle bon-F \rangle$ bɔ̃	* $\langle F \rangle$ ∅	* $\langle M \rangle$ ə	* $\langle bon-M \rangle$ bɔn	* $\langle bon-F \rangle$ bɔn	* $\langle F \rangle$ ə	* $\langle M \rangle$ ∅
 $ b\text{ɔ}n+\emptyset\#\text{akt}\text{ɛ}\text{ɛ} $					*			*
$ b\text{ɔ}n+\text{ə}\#\text{akt}\text{ɛ}\text{ɛ} $				*!	*			
$ b\text{ɔ}+\emptyset\#\text{akt}\text{ɛ}\text{ɛ} $	*!							*
$ b\text{ɔ}+\text{ə}\#\text{akt}\text{ɛ}\text{ɛ} $	*!			*				

Here, the lexical-phonological constraint * $\langle bon-M \rangle|b\text{ɔ̃}|$ militates against connecting the morpheme $\langle bon-M \rangle$ to the underlying form $|b\text{ɔ̃}|$, the lexical-phonological constraint * $\langle F \rangle|\emptyset|$ militates against connecting morphemic femininity to an underlying null form, and so on. In the serial account, the workings of the lexical constraints are quite simple: since they interact with nothing else, the winning underlying form of a morpheme is always the one that violates the lowest-ranked lexical constraint.

Now that the underlying form is known, the speaker connects it to the surface structure $/b\text{ɔ}.nak.t\text{ɛ}\text{ɛ}./$ in (3), where “.” denotes a syllable boundary and where the final underlying $|n|$ of $|b\text{ɔ}n|$ has been moved to the onset of the same syllable that contains the first two segments of *acteur* (a case of *liaison*).

(3) *Mapping the underlying form to a surface form*


$ b\text{ɔ}n+\emptyset\#\text{akt}\text{ɛ}\text{ɛ} $	* $ /ə/$	* $ ə / $	* $/n./$	* $/V.V/$	* $ ɔ̃ /ɔn/$	* $ ɔn /ɔ̃/$
 $/b\text{ɔ}.nak.t\text{ɛ}\text{ɛ}./$						
$/b\text{ɔ̃}.ak.t\text{ɛ}\text{ɛ}./$				*!		*
$/b\text{ɔ}.nə.ak.t\text{ɛ}\text{ɛ}./$	*!			*		
$/b\text{ɔ̃}.ə.ak.t\text{ɛ}\text{ɛ}./$	*!			**		*

Here we see four faithfulness constraints, in a generalized notation suitable for multilevel approaches: * $||/ə/$ militates against schwa insertion, * $|ə|/|$ against schwa deletion, * $|ɔ̃|/ɔn/$ against *n*-insertion, and * $|ɔn|/ɔ̃/$ against *n*-deletion. There are also two structural constraints: * $/n./$ against *n* in coda, and * $/V.V/$ against hiatus. Some of these constraints become relevant only in later tableaux.¹

¹ In this article, we ignore the potential multiplicity of syllabification candidates. Specifically, we assume that there are high-ranking constraints that dictate that *bonne maison* ‘good house’ is better represented as $/b\text{ɔ}n.m\text{ɛ}.z\text{ɔ̃}./$ than as $/b\text{ɔ}.nm\text{ɛ}.z\text{ɔ̃}./$, and, conversely, that *bon oiseau* ‘good bird’ is $/b\text{ɔ}.n\text{w}\text{a}.z\text{ɔ̃}./$ rather than $/b\text{ɔ}n.w\text{a}.z\text{ɔ̃}./$.

Finally, the speaker pronounces the surface form as the overt phonetic form [bɔnaktɛɪ] in (4), with the help of cue constraints (faithfulness-like constraints for phonetic implementation; Boersma 2007) and constraints against articulatory effort.

(4) *Mapping the surface form to a phonetic form*


/bɔ.nak.tɛɪ./	*/ɔ/[ɔn]	*/ɔn/[ɔ]	*[ə]	*/ə/[]	*/[ə]
 [bɔnaktɛɪ]					
[bɔaktɛɪ]		*!			
[bɔnəktɛɪ]			*!		*
[bɔəktɛɪ]		*!	*		*

Here, the cue constraint */ɔ/[ɔn] militates against pronouncing a phonological nasal vowel as a phonetic nasal consonant, and */[ə] militates against pronouncing a phonetic schwa without a phonological correspondent (the articulatory constraint *[ə] is needed below).


To summarize tableaux (1)–(4), one can now express the full route from meaning to sound as the winning *quintuplet* ‘good_i actor_i’ – ⟨bon-M; acteur_M⟩ – |bɔn+∅#aktɛɪ| – /bɔ.nak.tɛɪ./ – [bɔnaktɛɪ].

We now proceed to the word *mari* ‘husband’. While the word *acteur* is masculine and vowel-initial, *mari* is also masculine, but consonant-initial. The meaning ‘good_i husband_i’ shows up as [bɔmaksi], without any [n], the idea being that /n/ can phonologically show up only before vowel-initial forms such as /ak.tɛɪ./, and not before consonant-initial forms such as /ma.ɪi./. In a serial account, the first two mappings have no knowledge of this phonological conditioning, so that in (5) and (6) they operate the same way they did in the ‘good_i actor_i’ case.

(5) *Serial account: Mapping meaning to morphemes is insensitive to phonology*


‘good _i husband _i ’	*⟨FM⟩
 ⟨bon-M; mari _M ⟩	
⟨bon-F; mari _M ⟩	*!

(6) *Serial account: Mapping morphemes to an underlying form is insensitive to phonology*

⟨bon-M; mari _M ⟩	*⟨bon-M⟩ bɔ	*⟨bon-F⟩ bɔ	*⟨F⟩ ∅	*⟨M⟩ ə	*⟨bon-M⟩ bɔn	*⟨bon-F⟩ bɔn	*⟨F⟩ ə	*⟨M⟩ ∅
 bɔn+∅#maksi					*			*
bɔn+ə#maksi				*!	*			
bɔ+∅#maksi	*!							*
bɔ+ə#maksi	*!			*				


So the underlying form is $|\text{b}\bar{\text{o}}\text{n}+\emptyset\#\text{ma}\bar{\text{r}}\text{i}|$, with the same initial two morphemes as ‘good_i actor_i’, with the inclusion of an underlying $|\text{b}\bar{\text{o}}\text{n}|$. Now that phonological material is available, the coalescence of the vowel and the nasal ($\bar{\text{o}}\text{n} \rightarrow \bar{\text{o}}$) can enter the derivation. All authors mentioned above (Schane, Dell, Selkirk) agree that this is an early phonological rule. In OT, the phonological production phase can indeed enforce this change. The constraint against coda nasals ($*/\text{n}/$) forces underlying $|\text{b}\bar{\text{o}}\text{n}|$ to surface as $/\text{b}\bar{\text{o}}/$ in (7), thus violating a faithfulness constraint.

(7) *Serial account: Preconsonantal vowel-nasal coalescence in masculine forms must take place no earlier than in the phonology*

$ \text{b}\bar{\text{o}}\text{n}+\emptyset\#\text{ma}\bar{\text{r}}\text{i} $	$*/\bar{\text{o}}/$	$*/\bar{\text{e}}/$	$*/\text{n}/$	$*/\text{V.V}/$	$*/\bar{\text{o}}/ \bar{\text{o}}\text{n}/$	$*/\bar{\text{o}}\text{n}/ \bar{\text{o}}/$
$/\text{b}\bar{\text{o}}\text{n}.\text{ma}.\bar{\text{r}}\text{i}/$			*!			
 $/\text{b}\bar{\text{o}}.\text{ma}.\bar{\text{r}}\text{i}/$						*
$/\text{b}\bar{\text{o}}.\text{n}\bar{\text{e}}.\text{ma}.\bar{\text{r}}\text{i}/$	*!					
$/\text{b}\bar{\text{o}}.\bar{\text{e}}.\text{ma}.\bar{\text{r}}\text{i}/$	*!			*		*

Finally, the phonetic implementation phase in (8) offers no surprises, because there is a candidate that violates none of the constraints.


(8) *Serial account: Mapping the surface form to a phonetic form*

$/\text{b}\bar{\text{o}}.\text{ma}.\bar{\text{r}}\text{i}/$	$*/\bar{\text{o}}/[\bar{\text{o}}\text{n}]$	$*/\bar{\text{o}}\text{n}/[\bar{\text{o}}]$	$*/[\bar{\text{e}}]$	$*/\bar{\text{e}}/[\bar{\text{e}}]$	$*//[\bar{\text{e}}]$
$[\text{b}\bar{\text{o}}\text{nma}\bar{\text{r}}\text{i}]$	*!				
 $[\text{b}\bar{\text{o}}\text{ma}\bar{\text{r}}\text{i}]$					
$[\text{b}\bar{\text{o}}\text{n}\bar{\text{e}}\text{ma}\bar{\text{r}}\text{i}]$	*!		*		*
$[\text{b}\bar{\text{o}}\bar{\text{e}}\text{ma}\bar{\text{r}}\text{i}]$			*!		*

The whole quintuplet from meaning to sound can be summarized as ‘good_i husband_i’ – $\langle \text{bon-M}; \text{ma}\bar{\text{r}}\text{i}_\text{M} \rangle$ – $|\text{b}\bar{\text{o}}\text{n}+\emptyset\#\text{ma}\bar{\text{r}}\text{i}|$ – $/\text{b}\bar{\text{o}}.\text{ma}.\bar{\text{r}}\text{i}/$ – $[\text{b}\bar{\text{o}}\text{ma}\bar{\text{r}}\text{i}]$.


We are now left with the feminine case, where $[\text{n}]$ shows up despite a subsequent consonant, as in $[\text{b}\bar{\text{o}}\text{n}\text{vwa}\text{t}\bar{\text{y}}\text{b}]$ ‘good_i car_i’. As in the masculine case, the $*\langle \text{FM} \rangle$ constraint enforces gender agreement at the morpheme level (at least in the serial account).

(9) *Serial account: Mapping meaning to feminine morphemes*

‘good _i car _i ’	$*\langle \text{FM} \rangle$
$\langle \text{bon-M}; \text{voiture}_\text{F} \rangle$	*!
 $\langle \text{bon-F}; \text{voiture}_\text{F} \rangle$	


The early generative accounts mentioned above posit a schwa in the underlying form.

(10) *Serial account: Underlying feminine forms have schwa*

$\langle \text{bon-F}; \text{voiture}_F \rangle$	* $\langle \text{bon-M} \rangle$ bɔ̃	* $\langle \text{bon-F} \rangle$ bɔ̃	* $\langle \text{F} \rangle$ ∅	* $\langle \text{M} \rangle$ ə	* $\langle \text{bon-M} \rangle$ bɔn	* $\langle \text{bon-F} \rangle$ bɔn	* $\langle \text{F} \rangle$ ə	* $\langle \text{M} \rangle$ ∅
bɔn+∅#vwa.tyʁ			*!			*		
 bɔn+ə#vwa.tyʁ						*	*	
bɔ̃+∅#vwa.tyʁ		*!	*					
bɔ̃+ə#vwa.tyʁ		*!					*	


The existence of schwa prevents the deletion of /n/, because /n/ is not in coda position.

(11) *Serial account: Schwa shows up in feminine forms*

bɔn+ə#vwa.tyʁ	* /ə/	* ə /	*/n./	*/V.V/	* ɔ̃ /ɔn/	* ɔn /ɔ̃/
/bɔn.vwa.tyʁ./		*!	*			
/bɔ̃.vwa.tyʁ./		*!				*
 /bɔ.nə.vwa.tyʁ./						
/bɔ̃.ə.vwa.tyʁ./				*!		*

In these serial generative accounts, schwa is dropped later in the derivation, that is, after coda *n*-deletion. In an OT account with only two phonological levels, as here, the dropping of schwa must be relegated to the phonetic implementation, as seen in (12).

(12) *Serial account: Schwa drop in phonetic implementation*

/bɔ.nə.vwa.tyʁ./	* ɔ̃ /ɔn]	* ɔn / ɔ̃	*[ə]	* ə /[]	* / / ə]
 [bɔnvwa.tyʁ]				*	
[bɔ̃vwa.tyʁ]		*!		*	
[bɔnəvwa.tyʁ]			*!		
[bɔ̃əvwa.tyʁ]		*!	*		

Note the crucial ranking of *|[ə]| over *|ə|/[]; that is, it is worse to have a schwa in the phonetics than to drop a phonological schwa from the phonetics.

A source of complexity in this serial analysis is its crucial reliance on the existence of two phonological and/or phonetic mappings. In the original rule-ordering approach, the rule of schwa deletion had to be ordered after the rule of vowel-nasal coalescence. In OT, such a situation of counterfeeding interaction cannot be modeled with a single mapping, if the two processes (here, schwa deletion and vowel-nasal coalescence) are governed by separate faithfulness constraints

(here, $*|\partial//$ and $*|\partial n|/\delta/$; Kirchner 1995, Orgun 1995, Smolensky 1995, Gnanadesikan 1997, Moreton and Smolensky 2002). In our French example, there is no ranking of the constraints in (11) that yields schwa deletion in the phonology proper; that is, it is impossible to derive a phonological $/b\partial n.vwa.ty\epsilon./$ with the given constraints and levels. With a high-ranked $*|\partial/$ in (11), $/b\delta.vwa.ty\epsilon./$ would win, because $*|n./$ outranks $*|\partial n|/\delta/$, a ranking that is crucial to make (7) work. In other words, no ranking of $*|\partial/$, $*|n./$, and $*|\partial n|/\delta/$ will produce both vowel-nasal coalescence in $/b\delta.ma.bi./$ and the surfacing of the $/n/$ in $/b\partial n.vwa.ty\epsilon./$, and schwa drop can only take place in the “later” phonetic implementation phase. If one insists on having schwa deletion in the phonology instead, perhaps because other phonological rules interact with it (e.g., Dell 1973:188), one will need an intermediate phonological level of representation, such as the *word level* proposed by theories of lexical phonology (Kiparsky 1982, Bermúdez-Otero 2003; schwa drop would then take place in the postlexical phonology).² The conclusion is that given our straightforward constraint set, the opacity of the interaction between vowel-nasal coalescence and schwa drop cannot be handled with only two levels of phonological and/or phonetic representation, and that it *can* be handled with three levels, as it is here.

2.2 The Parallel Analysis of Gender Selection and Liaison

A parallel account may look entirely differently at the matter than the serial account. According to Encrevé (1988), Encrevé-Lambert (1971) proposed that French speakers opt for *gender disagreement* if this benefits the phonology: the feminine phrase *l'idée* ‘the idea’ has (a reduced form of) the masculine article *le* instead of the feminine article *la*, because the schwa of *le* ($|l\partial|$), but not the full vowel of *la* ($|la|$), can be deleted before vowels ($/li.de./$); *un vieil acteur* ‘an old actor’ has the feminine adjective *vieille* rather than the masculine adjective *vieux*, because *vieille* ($|vj\epsilon j+\partial|$) can provide an onset to *acteur* ($|.vj\epsilon.jak.t\epsilon\epsilon.|$) whereas *vieux* ($|vj\partial+\emptyset|$) cannot; *mon idée* ‘my idea’ has the masculine possessive pronoun *mon* rather than the feminine *ma*, because *mon* ($|m\delta(n)|$) can provide an onset to *idée* ($/m\delta.ni.de./$) whereas *ma* ($|ma|$) cannot.


Translating this idea to OT, Tranel (1996) proposed the gender agreement constraint that we write here as $\langle FM \rangle$, and had it interact with phonological constraints equivalent to $*|n./$ and $\langle V.V \rangle$. While Tranel made no attempt to formalize this situation with more than two levels of representation, it can straightforwardly be formulated in a principled manner within the model of figure 1, where the morphosyntactic constraint of gender disagreement and the phonological-structure constraints of syllable onsets and codas play their roles at different levels of representation. Crucially, a constraint at a “later” level of representation (phonological surface form) dominates a constraint at an “earlier” level of representation (morphemes), a situation that can only occur if the levels are handled in parallel (or *interactively*) rather than serially.

In the present example, we have the option of generalizing the *ma* ~ *mon* alternation to *bon* ~ *bonne*, regarding the latter pair as suppletive and ignoring any phonological relationship. Under that view, French speakers wanting to produce the meaning ‘good actor’ have the option of

² Or one could introduce another source of complexity and add a high-ranked conjoined faithfulness constraint along the lines proposed by Smolensky (1995), that is, $*|\partial n|/\delta/ \& *|\partial//$.

choosing the morpheme $\langle bon-F \rangle$ instead of $\langle bon-M \rangle$. The resulting morpheme sequence $\langle bon-F; acteur_M \rangle$ does violate the morphosyntactic constraint against gender disagreement between adjective and noun, but does have the advantage that $\langle bon-M \rangle$ can take the underlying form $[b\delta]$, violating no faithfulness constraints in $/b\delta.ma.ki./$, and $\langle bon-F \rangle$ can take the underlying form $[b\delta]$, violating no faithfulness constraints in $/b\delta.n.vwa.ty\epsilon./$ or $/b\delta.nak.t\epsilon\epsilon./$. The tableau in (13) shows how the constraint against hiatus in the phonological surface form can force the selection of the feminine morpheme $\langle bon-F \rangle$ for the masculine noun $\langle acteur_M \rangle$.

(13) *Parallel account: Phonology influences morphology*

'good _i actor _i '	* $\langle bon-M \rangle$ [b\delta]	* $[\delta]/\delta n/$	*/V.V/	* $\langle FM \rangle$	* $[\delta n]/\delta/$	* $\langle bon-F \rangle$ [b\delta]
$\langle bon-M; acteur_M \rangle$ [b\delta + \emptyset # akt\epsilon\epsilon] $/b\delta.nak.t\epsilon\epsilon./$ [b\delta nakt\epsilon\epsilon]	*!					
$\langle bon-M; acteur_M \rangle$ [b\delta n + \emptyset # akt\epsilon\epsilon] $/b\delta.ak.t\epsilon\epsilon./$ [b\delta\delta akt\epsilon\epsilon]	*!		*		*	
$\langle bon-M; acteur_M \rangle$ [b\delta + \emptyset # akt\epsilon\epsilon] $/b\delta.nak.t\epsilon\epsilon./$ [b\delta nakt\epsilon\epsilon]		*!				
$\langle bon-M; acteur_M \rangle$ [b\delta + \emptyset # akt\epsilon\epsilon] $/b\delta.ak.t\epsilon\epsilon./$ [b\delta\delta akt\epsilon\epsilon]			*!			
 $\langle bon-F; acteur_M \rangle$ [b\delta n + \emptyset # akt\epsilon\epsilon] $/b\delta.nak.t\epsilon\epsilon./$ [b\delta nakt\epsilon\epsilon]				*		*

The working of the morphosyntactic constraint * $\langle FM \rangle$ is no longer trivial, as it was in section 2.1, and the workings of the lexical-phonological constraints are no longer simple, as they were in section 2.1. Instead, the morphosyntactic constraint and the lexical-phonological constraints now interact in interesting ways with constraints at a ‘later’ level, namely, faithfulness constraints and a structural constraint.

When we compare the serial account of section 2.1 with the parallel account of section 2.2, we see several differences. In the parallel account, surface ‘ghost’ (i.e., unpronounced) schwas

as in (11) are no longer needed, even for feminine forms. Instead, there is suppletive allomorphy at the underlying level: [bɔ̃] is the masculine form, [bɔ̃n] the feminine form. In ‘good_i actor_i’, an additional allomorphy on the basis of gender takes place: the feminine form is selected because a gender change (violating *⟨FM⟩) is less bad than hiatus (violating */V.V/). To sum up, the parallel analysis gets rid of the ghost segment /ə/, and no longer do any nontrivial processes have to take place in phonetic implementation; these two advantages (and the advantage of being able to simultaneously account for the *ma* ~ *mon* suppletion) come at the cost of losing any generalization about a phonological relationship between /bɔ̃/ and /bɔ̃n/.

Another apparent disadvantage of the parallel model is the gigantic size of a full list of candidates with their violations. For the sake of brevity, tableau (13) has been reduced to the bare minimum number of constraints and candidates: it includes only 6 of the 20 constraints of section 2.1, assumes a trivial relation between surface and phonetic form, and otherwise excludes many potentially relevant candidate quadruplets. A full version of tableau (13) would contain hundreds of candidates and thousands of violation marks. This advantage, however, is only apparent: in section 2.3, we introduce the ‘candidate graph,’ a method that captures all the possible candidates in a much more concise and visually informative manner.

2.3 Candidate Graphs

Each of the three winning candidates for the serial analysis in section 2.1, as well as each of the three winning candidates for the parallel analysis in section 2.2, can be visualized as a path through the *candidate graph* in figure 2. In figure 2, each level of representation is represented as a number of nodes stacked vertically; every possible candidate is represented as a path that runs from the left end of the graph to the right end, visiting exactly one node at each level. We can therefore call each possible quintuplet a *candidate path*, or just *path*. The three winning paths of the parallel analysis of section 2.2, for instance, are drawn in figure 2 as thick lines.

The parallel analysis benefits greatly from visualizing it with candidate graphs. Given the meaning ‘good_i actor_i’, for instance, the speaker can choose from $2 \times 4 \times 4 \times 4 = 128$ paths to the phonetic form, at least in the simplified world of figure 2. A full version of tableau (13) would therefore have 128 quadruplets as candidates. In comparison, the graph in figure 2 contains only 14 single forms (besides the input form), rather than the 512 forms that 128 quadruplets would require in the full version of tableau (13). This graph representation constitutes a huge gain in readability, but also a large gain in the computation speed for evaluations as well as for learning, as the following sections show.

3 Efficient Evaluation and Learning

This section discusses how OT evaluation and learning can be done with candidate graphs.

3.1 Constraints in the Graph

We have seen in figure 2 how candidates in a multilevel OT grammar can be described as paths along a directed (left-to-right) graph, where each *node* represents a form on some level of representation, and each *connection* (the graph-theoretical term is *edge*) represents a possible

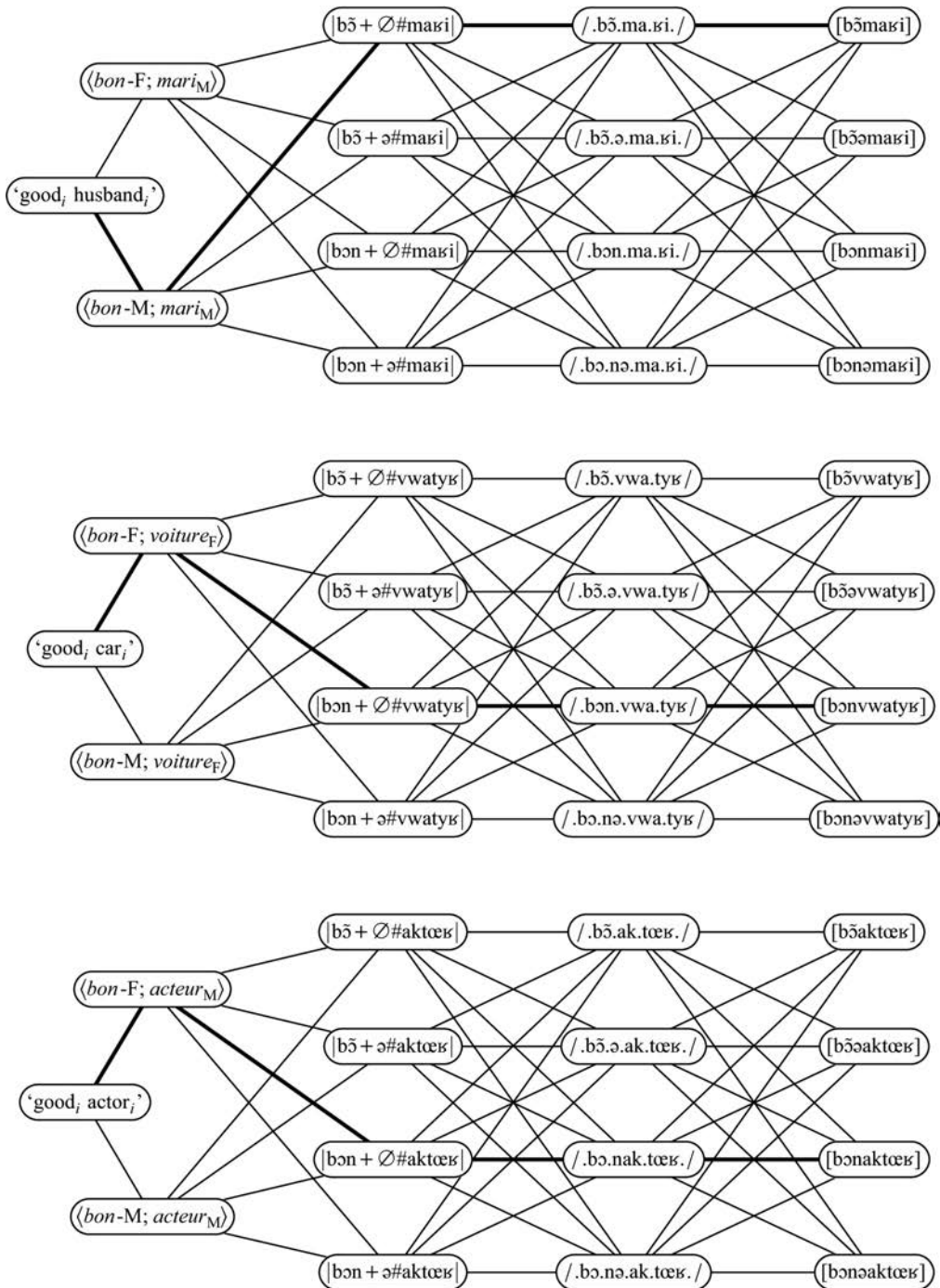


Figure 2

Some French graphs of representations. Each path from left to right is a candidate production. The thick paths illustrate the parallel gender allomorphy analysis of section 2.2.

mapping between two forms on adjacent levels of representation. Each path in figure 2 visits five nodes and follows four connections. The graphs can visualize not only candidates, but also constraint violations: the intralevel constraints of figure 1 and section 2 militate against visiting some nodes, and the interlevel constraints of figure 1 and section 2 militate against following some connections.

For illustrating how evaluation and learning work, we take a simplified version of our French grammar, containing a smaller number of constraints and forms (e.g., only two phonetic forms instead of four). The constraints are indicated below by numerical labels ① through ⑩.

①	*⟨bon-M⟩ bɔn	lexical-phonological constraint
②	* ɔ̃ [ɔn]	cue constraint
③	*⟨FM⟩	morphosyntactic constraint
④	*⟨bon-M⟩ bɔ̃	lexical-phonological constraint
⑤	* ɔ̃ /ɔn/	faithfulness constraint
⑥	*⟨bon-F⟩ bɔn	lexical-phonological constraint
⑦	*⟨bon-F⟩ bɔ̃	lexical-phonological constraint
⑧	* ɔn /ɔ̃/	faithfulness constraint
⑨	* ɔn [ɔ̃]	cue constraint
⑩	* V.V	structural constraint

Figure 3 shows the graph for the meaning ‘good_i actor_i’. The two intralevel constraints *⟨FM⟩ and */V.V/ are indicated by labels ③ and ⑩ placed on the relevant nodes, while the remaining (interlevel) constraints are indicated by labels placed on the relevant edges.

The graph contains 16 paths that run from the meaning at the left of the graph toward the phonetic forms at the right. One of these 16 paths is the optimal candidate under the given constraint ranking. The next section explains how the search for the optimal candidate (the *evaluation*) proceeds.

3.2 Efficient Evaluation: The Elimination Version

The optimal candidate in a graph is the path whose nodes and edges incur the least serious violations. Analogously to the familiar procedure for evaluating candidates in tableaux (Prince and Smolensky 1993), there exists a procedure for graphs that eliminates candidates by iterating

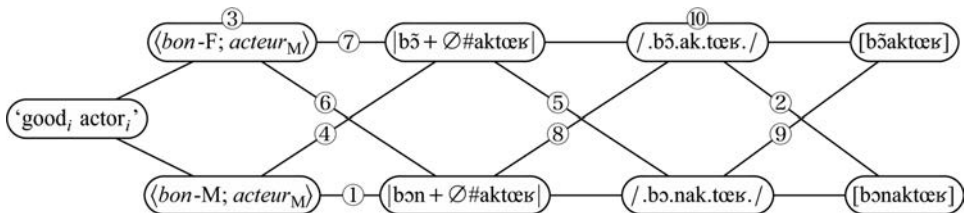


Figure 3

The candidate graph for the meaning ‘good_i actor_i’. Also the starting point for the evaluation procedure of section 3.2.

through the constraint hierarchy, starting with the highest-ranked constraint. This slightly informal procedure is presented in section 3.2.1, while section 3.2.2 presents a more formal account in terms of dynamic programming.

3.2.1 Evaluation by Elimination of Edges and Nodes The first evaluation method for candidate graphs that we discuss is analogous to the usual tableau evaluation. At the start, all nodes and edges in the graph are “active”; that is, they can potentially contribute to the optimal path. Next, nodes and edges are iteratively eliminated in an order dictated by the constraint hierarchy, until one or more optimal paths are left.

The procedure starts by considering the highest-ranked constraint. The edges or nodes associated with the constraint are tentatively deactivated. Next, the algorithm checks how many paths are left, that is, via how many routes the right side of the graph can be reached from the left side (this can be done efficiently, with an algorithm that involves each edge only once).³ If the number of remaining paths is zero, we must conclude that all the candidates are apparently “tied” on the constraint; the tentative deactivation is then undone, and the algorithm proceeds to the next constraint. In the other case (i.e., if the number of remaining paths is greater than zero), the deactivation of the nodes or edges is made permanent (for the duration of this evaluation); under the stipulation that paths cannot follow deactivated edges and cannot visit deactivated nodes, this step eliminates all candidate paths that fatally violate the constraint. If the number of remaining paths is one, we must conclude that this sole remaining path represents the optimal candidate, and the algorithm terminates successfully. Finally, if multiple paths remain, the above steps are repeated with the next-highest-ranked constraint. These iterations are repeated down the constraint hierarchy, until either a single path remains or the lowest-ranked constraint has been handled. Should there still be multiple paths through the graph after all constraints have been handled, we must conclude that all these remaining paths are optimal candidates.

Figures 4–9 take us stepwise through the elimination procedure for the case of figure 3. The set of possible candidates at the start of evaluation is defined by figure 3. The constraints of

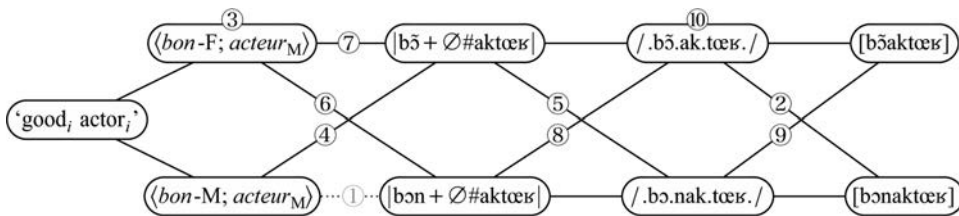


Figure 4
The graph after handling constraint ①; 12 out of 16 paths are left.

³ The number of paths can be computed iteratively from left to right: for a node X on level n , the number of paths that lead to it from the single node on level 1 is zero if the node is deactivated, and otherwise it is the sum of the numbers of paths to those nodes on level $n-1$ for which the edges to node X are not deactivated. The complexity of counting the number of paths from the left end to the right end of the graph is therefore linear in the number of local connections (if we disregard the logarithmic complexity of addition).

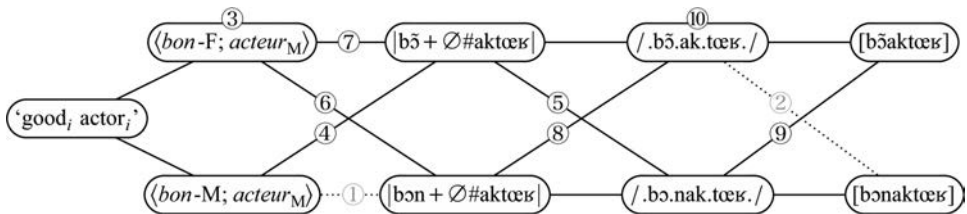


Figure 5
The graph after handling constraint ②; 9 paths are left.

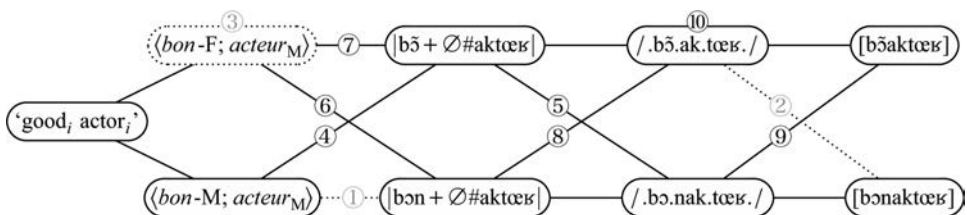


Figure 6
The graph after handling constraint ③; 3 paths are left.

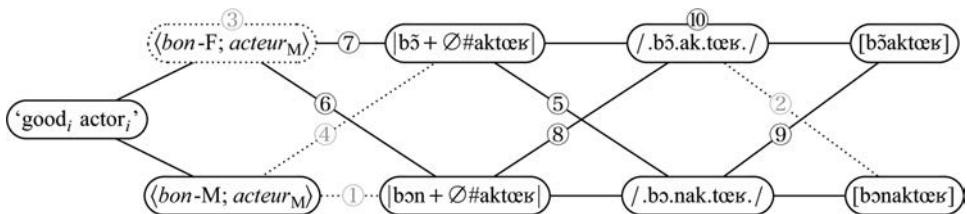


Figure 7
The graph after trying to handle constraint ④; no paths are left.

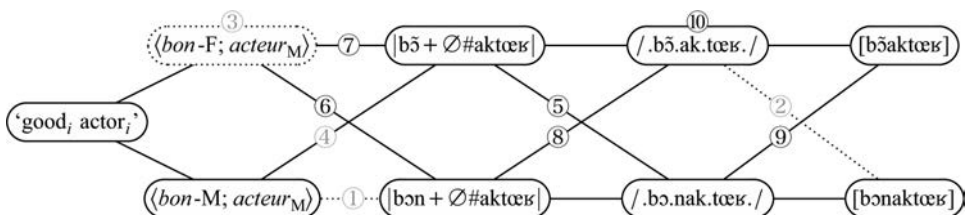


Figure 8
The graph after fully handling constraint ④; again, 3 paths are left.

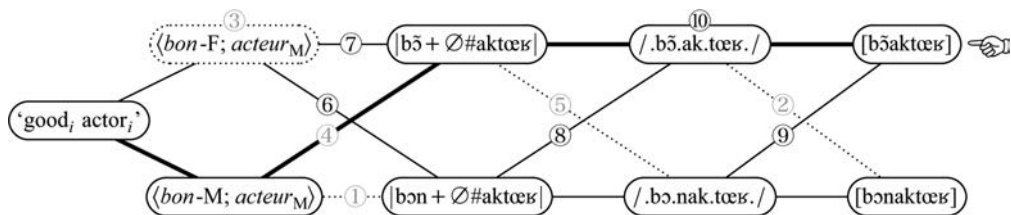


Figure 9
The graph after handling constraint ⑤; a single path (thick lines; pointing finger) is left.

section 3.1 are ranked from high to low by their label, that is, with $*(\text{bon-M})|\text{b}\text{ɔ}\text{n}|$ ranked highest and $*/\text{V.V}/$ lowest. The first step, now, is that the highest-ranked constraint, constraint ①, tentatively deactivates the connection between the morpheme sequence $\langle \text{bon-M}; \text{acteur}_M \rangle$ and the underlying form $|\text{b}\text{ɔ}\text{n}+\emptyset\#\text{akt}\text{ɛ}\text{ɾ}|$, which tentatively eliminates all 4 paths that travel this connection. As this still leaves a number of paths from left to right through the graph (namely, 12), the deactivation is made “permanent.” Figure 4 depicts this elimination by using dotted lines. In the next iteration (figure 5), constraint ② severs the connection between the surface form $/\text{b}\text{ɔ}. \text{ak}. \text{t}\text{ɛ}\text{ɾ}/$ and the phonetic form $[\text{b}\text{ɔ}\text{n}\text{akt}\text{ɛ}\text{ɾ}]$. Next, the intralevel constraint ③ deactivates the morpheme sequence $\langle \text{bon-F}; \text{acteur}_M \rangle$, eliminating all paths that visit this node (figure 6). After this, only 3 paths remain out of the original 16. In figure 7, constraint ④ tentatively deactivates the connection between $\langle \text{bon-M}; \text{acteur}_M \rangle$ and $|\text{b}\text{ɔ}+\emptyset\#\text{akt}\text{ɛ}\text{ɾ}|$. As this step reduces the number of paths to zero (one can no longer get from left to right through the graph in figure 7), this connection is reactivated, as seen in figure 8 (the lines become solid again, but constraint ④ stays grayed out). In figure 9, constraint ⑤ eliminates the connection between $|\text{b}\text{ɔ}+\emptyset\#\text{akt}\text{ɛ}\text{ɾ}|$ and $/\text{b}\text{ɔ}. \text{n}\text{ak}. \text{t}\text{ɛ}\text{ɾ}/$, and a single path remains. This optimal path corresponds to the candidate ‘good_i actor_i’ – $\langle \text{bon-M}; \text{acteur}_M \rangle - |\text{b}\text{ɔ}+\emptyset\#\text{akt}\text{ɛ}\text{ɾ}| - / \text{b}\text{ɔ}. \text{ak}. \text{t}\text{ɛ}\text{ɾ} / - [\text{b}\text{ɔ}\text{akt}\text{ɛ}\text{ɾ}]$ and is marked by thick lines in the figure.

The algorithm is much faster than a full search of all candidates would be, especially for more realistic numbers of constraints or amounts of data than contained in the toy example of this section. The complexity of the algorithm is linear in the number of constraints (the outer loop), linear in the maximum number of violations per constraint (the inner loop), and linear in the number of edges plus nodes (for scanning the locations of violation, and for counting the number of paths according to footnote 3); the complexity therefore no longer depends exponentially on the number of levels of representation, as it did in Apoussidou’s (2007) and Boersma’s (2008) simulations.

3.2.2 Evaluation by Dynamic Programming Computationally, the overall structure of the graph (a *trellis*) invites a practical implementation in terms of a *dynamic programming* algorithm, whose complexity is linear in the number of edges and nodes. Dynamic programming computes the “cheapest” path from the input to (in our case) the right side of the graph. While originally designed for additive cost functions (Ford 1956, Bellman 1957, Dijkstra 1959) or multiplicative probabilities (Viterbi 1967), dynamic programming applies to OT’s lexicographic constraint order as well (Riggle 2004:159–161, 2009).

To start off, we associate each edge and each node with a *cost*, which is a vector of constraint violations, sorted by ranking. For instance, in figure 4 the cost of the node $\langle \text{bon-F}; \text{acteur}_M \rangle$ is (0, 0, 1, 0, 0, 0, 0, 0, 0), because this node violates only the third-ranked constraint $*(\text{FM})$, and the cost of the edge from $|\text{b}\text{ɔ}\text{n}+\emptyset\#\text{akt}\text{ɛ}\text{ɾ}|$ to $/\text{b}\text{ɔ}. \text{ak}. \text{t}\text{ɛ}\text{ɾ} /$ is (0, 0, 0, 0, 0, 0, 0, 1, 0, 0), because this edge violates only the eighth-ranked constraint $*|\text{ɔ}\text{n}|/\text{ɔ}/$. Costs can be *added*, which goes element by element; for example, if the cost *A* equals (1, 0, 2) and the cost *B* equals (1, 1, 0), then *A* + *B* equals (2, 1, 2). Finally, costs can be *compared* (OT’s *lexicographic ordering*: Prince and Smolensky 1991) by checking the first element (counted from the left) in which they differ:

in the example of the previous sentence, A is less than B , because the first element in which A and B differ is their second element, which is smaller in A (namely, 0) than in B (namely, 1). As a result, we can also talk about the *minimum* of a *set* of costs, a notion we need in the algorithm. As Riggle (2009) shows, the existence of well-defined addition and “minimum” operations is precisely what makes OT suitable for dynamic programming.

In our formalization, we number the levels, and within the levels we number the nodes. The input level is level 0; its number of nodes is $N_0 = 1$. Then there follow L (in our example: 4) levels of representation, each with N_l nodes ($l = 1 \dots L$). Each node n at level l comes with a cost $nodeCost [l, n]$, where n runs from 1 to N_l , and each connection from node m at level $l-1$ to node n at level l comes with a cost $edgeCost [l, m, n]$. At the right edge, there is a single invisible node at level $L+1$, which is connected to each node at the phonetic level without violating any constraints ($N_{L+1} = 1$). Here is the pseudocode for the initialization of the relevant variables in the algorithm (the scope of **for** blocks is determined by indentation):

```

nodeCost [0, 1] := the violation vector of the input node, sorted by constraint ranking
for  $l = 1 \dots L$ 
  for  $n = 1 \dots N_l$ 
    for  $m = 1 \dots N_{l-1}$ 
      edgeCost [ $l, m, n$ ] := the violation vector of edge $_{l,m,n}$ , sorted by constraint ranking
      nodeCost [ $l, n$ ] := the violation vector of node $_{l,n}$ , sorted by constraint ranking
    for  $m = 1 \dots N_L$ 
      edgeCost [ $L+1, m, 1$ ] := (0, 0, ...) # a vector of  $C$  zeroes
      nodeCost [ $L+1, 1$ ] := (0, 0, ...)

```

After this initialization, we employ a dynamic programming algorithm that takes into account the possibility of tied candidates (i.e., equally good paths). In left-to-right graphs like the ones in this article, dynamic programming works iteratively from left to right, in the following way. Suppose that for each node Y on level $n-1$ the best paths (and their cost) for going from the input node (the single node on level 0) to Y is known. The best paths for going from the input node to a node X on level n are then the paths through that node Y for which the cost to Y plus the cost of going from Y to X is less than (or equal to) the cost for all other nodes on level $n-1$. Since the best path is trivial for all nodes on level 1, and the best path can be computed for level n if it is known for level $n-1$, the best path can be computed for all nodes at all levels, including for the single invisible node at the last level. The algorithm uses the minimum operation (“min”) and the addition operation (“+”) for the cost vectors, as follows:

```

nodeRoutes [0, 1] := 1 # the number of best routes to the input node
nodePathCost [0, 1] := nodeCost [0, 1]
for  $l = 1 \dots L+1$ 
  for  $n = 1 \dots N_l$ 
    for  $m = 1 \dots N_{l-1}$ 
      edgePathCost [ $m$ ] := nodePathCost [ $l-1, m$ ] + edgeCost [ $l, m, n$ ]

```

```

minimumEdgePathCost :=  $\min_{m=1}^{N_{l-1}}$  edgePathCost [m]
for m = 1 . . .  $N_{l-1}$ 
  if edgePathCost [m] = minimumEdgePathCost
    edgeRoutes [l, m, n] := nodeRoutes [l-1, m]
  else
    edgeRoutes [l, m, n] := 0
  nodeRoutes [l, n] :=  $\sum_{m=1}^{N_{l-1}}$  edgeRoutes [l, m, n]
  nodePathCost [l, n] := minimumEdgePathCost + nodeCost [l, n]

```

After this, $\text{edgeRoutes } [l, m, n]$ contains the number of best paths from the input node to node n at level l that go through node m at level $l-1$, and $\text{nodeRoutes } [l, n]$ contains the number of best paths from the input node to node n at level l .

Now that we know the number of best routes to any node, including the rightmost (invisible) node, we can randomly choose an optimal path back from that rightmost node to the input node, by iteratively choosing edges from right to left, with probabilities proportional to the number of best routes along the edges at each level, as follows:

```

optimalNode [L+1] := 1 # the invisible node at the end
for l = L . . . 1 # backtracking, i.e., counting down
  chosenRoute := randomInteger (1, nodeRoutes [l+1, optimalNode [l+1]])
  node := 0
  routes := 0
  while routes < chosenRoute
    node := node + 1
    routes := routes + edgeRoutes [l+1, node, optimalNode [l+1]]
  optimalNode [l] := node

```

Here, $\text{randomInteger } (a, b)$ is a function that chooses a whole number between a and b (inclusively), all with equal probability. When the algorithm finishes, the optimal path is given by $\text{optimalNode } [1 \dots L]$; for instance, in figure 9 the optimal path is given by $\text{optimalNode } [1] = 2$, $\text{optimalNode } [2] = 1$, $\text{optimalNode } [3] = 1$, and $\text{optimalNode } [4] = 1$, at least if at each level of representation the nodes are numbered from top to bottom.

3.2.3 Range of Application The informal account of section 3.2.1 and the more formal account of section 3.2.2 are equivalent: basically, the former travels the constraint hierarchy in the outer loop, whereas the latter does so in the inner loop (in the addition and in the comparison function).

For the algorithm to work, the constraints have to honor a strong restriction, namely, that each constraint evaluates either nodes on a single level or connections between adjacent levels. For instance, a constraint that is violated only for paths that follow a specific underlying form node *and* a specific surface form node is not allowed, and a constraint that is violated only for a specific connection from (e.g.) underlying form through surface form to phonetic form is not allowed either. All constraints proposed within the early version of OT that just map underlying

form to surface form automatically satisfy this restriction, and all the constraints mentioned here in section 2 do so as well.⁴

There is no limitation on the number of violations per constraint. In the informal account of section 3.2.1, we tacitly assumed that every constraint could be violated only once (although the account can easily be extended by considering a doubly violated constraint as a constraint that is ranked slightly higher than the same constraint if singly violated), but the formal account of section 3.2.2 makes clear that no such assumption is needed.

The algorithm is compatible not only with the notion of tied *paths* (see section 3.2.2), but also with the notion of crucially tied *constraints* (Prince and Smolensky 1993:fn. 31, Anttila 1997, Tesar and Smolensky 1998:241). Both in the informal account of section 3.2.1 and in the formal account of section 3.2.2, two constraints that are ranked at the exact same height can be collapsed, before evaluation starts, into a single constraint. However, our simulations exclusively use Stochastic OT (Boersma 1998), in which each constraint's ranking is numeric and contains a bit of noise, so that constraint ties have probability zero of occurring.

In the French example, given an input, all nodes are exhaustively connected to all nodes at the next level. This is not a real restriction: to allow unconnected nodes between consecutive levels, one can set the relevant *edgeCost* to a vector of positive infinity in the algorithm of section 3.2.2, or alternatively, the algorithm can easily be adapted to visit only the connected nodes. The efficiency of the algorithm, as compared with the efficiency of enumerating all possible paths, then depends on the degree to which nodes tend to be connected to multiple nodes at the previous level: in the worst case (i.e., if each noninput node is connected to only one node at the previous level), big-list evaluation and graph evaluation are equally efficient; in the best case (i.e., if each node is connected to all nodes at the previous level, as in our French case), the efficiency gain of graph evaluation over big-list evaluation is maximal.

3.3 *Efficient Learning from Form-Meaning Pairs*

The reader may have noticed that the winning phonetic form [bɔ̃aktœʁ] in figure 9 is not correct French. The inclusion of this failure here is deliberate: it highlights the fact that correct French is not something that children start doing automatically when they are born into French-speaking families; instead, they have to *learn* to produce French, presumably from the language input they receive from their environment. In this section, we describe how a learner can process French data in such a way that she does come to produce correct French.

What a learner of French hears is overt phonetic utterances. If we assume that the learner is capable of inferring the meaning of these utterances, we can say that the learner obtains knowl-

⁴ The restriction of adjacent levels can be lifted (as long as the left-to-right directedness continues to be honored). If the graph has (e.g.) direct connections between underlying form and phonetic form, Dijkstra's (1959) algorithm, which is more general than Viterbi's (1967) algorithm employed here, can do the trick; the informal method of section 3.2.1 would also still work.

edge of *pairs* of meaning and phonetic form. For our toy French example, the relevant learning data then consist of the following form-meaning pairs:

- [bɔ̃nvwatyɛ] paired with ‘good_i car_i’
- [bɔ̃naktœɛ] paired with ‘good_i actor_i’
- [bɔ̃makɪ] paired with ‘good_i husband_i’

We have shown in figures 3–9 how the learner computes an optimal phonetic form by eliminating connections and forms from the graph until a single candidate path from meaning to phonetic form remains. However, we may assume that a learner does not start out with a constraint hierarchy that will pair every meaning to the correct phonetic form; instead, the OT learning literature has proposed that either all constraints start out being ranked at the same height (Tesar and Smolensky 1993) or all constraints on nodes outrank all constraints on connections (e.g., ‘markedness >> faithfulness’: Levelt 1995). Following the OT literature on learning from overt forms (Tesar and Smolensky 1998, 2000, Apoussidou and Boersma 2004, Biró 2013, Jarosz 2013a), we regard it as the learner’s job in our French example to rearrange the constraint hierarchy until the phonetic form produced for each of the three meanings corresponds to the form presented in the learning data.

Following the idea of error detection in OT learning (Tesar and Smolensky 1998), the learner can learn by comparing two paths: the path that she would produce herself, which for our example is simply the path in figure 9, and the path that she considers correct. We will now explain in detail first how the learner determines the ‘correct’ path, and then how she learns from it.

To determine the ‘correct’ path, the learner follows a procedure that generalizes the idea of *robust interpretive parsing* (Tesar and Smolensky 2000). When given a form-meaning pair, the learner determines which of the many possible paths from the meaning to the *given* (correct) phonetic form satisfies the constraints best—that is, the path between given meaning and form that is optimal according to the learner’s current constraint ranking. This is done by a procedure analogous to the one for evaluation in section 3.2, with one important additional step at the beginning, namely, the deactivation of all phonetic forms that are not given in the data. Figure 10 illustrates the procedure with the pair ‘good_i actor_i’ ~ [bɔ̃naktœɛ]. The first step is that all phonetic forms except [bɔ̃naktœɛ] are deactivated; since our example has only one such form, [bɔ̃aktœɛ], this step deactivates only the node [bɔ̃aktœɛ], as can be seen in figure 10a.⁵ Typically, after this initial deactivation, there remain many possible paths through the graph (in figure 10a, there are eight). To find the optimal path from among these, the learner follows the exact same efficient procedure as in section 3.2, namely, deactivating nodes and connections in an order determined by the constraint ranking: constraints ① and ② sever two connections, and constraint ③ deactivates a node, as illustrated in figure 10b. As with the evaluation procedure in section 3.2, only a single path is left (or, in the general case, a few equally optimal paths). Because of the initial deactivation of non-French phonetic forms, this path must run from the meaning given in the data to the phonetic form given in the data. In the figure, this path is drawn with thick

⁵ In later sections, we apply robust interpretive parsing to the larger graphs of figure 2. In those cases, this first step deactivates three phonetic nodes at once.

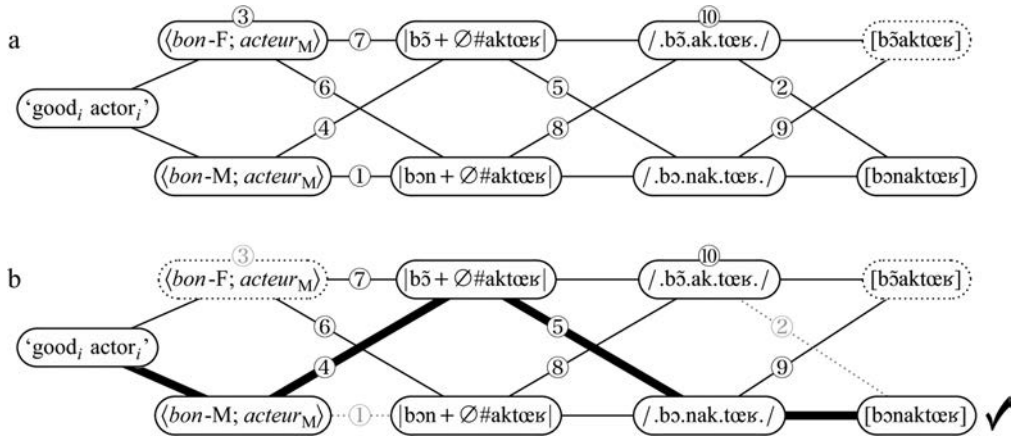


Figure 10

Robust interpretive parsing: initial deactivation of non-French phonetic forms, followed by the elimination of edges and nodes by the constraint hierarchy. One path remains (very thick lines).

lines and provided with a check mark. The learner now regards the path of figure 10 as the *correct path*, at least for the purpose of handling the current form-meaning pair.

Now that the learner knows the “correct” path, she proceeds to learn from it, that is, to potentially change her constraint ranking. To achieve this, she starts by comparing the “correct” path of figure 10b with the produced path of figure 9. If the two paths are the same, the learner sees no reason to change her grammar (and indeed, her produced phonetic form is correct French). If the two paths are different, the learner decides that she made an *error* and that her grammar has to change.⁶ Analogously to learning from the violation profiles of one “winner” and one “loser” candidate in an OT tableau (Tesar and Smolensky 1998), the learner initiates the change by comparing the violation profile of her produced path with that of the “correct” path, as in figure 11. She then updates her grammar by applying one of several OT update rules that have

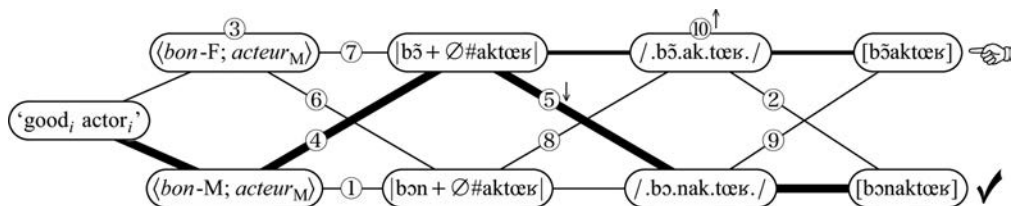


Figure 11

Learning by comparing the produced path of figure 9 with the “correct” path of figure 10b.

⁶ This is appropriate, because if the two paths are different, their phonetic forms must also be different. Intuitively, this must be true because if the phonetic forms of the two paths are identical, the paths must be identical as well (Bellman’s (1957) principle of optimality: if the overall optimal path contains phonetic form X, then this optimal path must be in the set of all possible paths to phonetic form X, and it must be the best of those paths).

been proposed in the literature: following the procedure used by Boersma (1998, 2008) and Apoussidou (2007), the rankings of the constraints violated by the production path are raised, and the rankings of all the constraints violated by the “correct” path are lowered. In figure 11, the two paths differ in the right half of the figure, so that constraint ⑤ has to be lowered and constraint ⑩ has to be raised, as indicated by the arrows.

We have found efficient procedures for evaluation (section 3.2) and for learning (section 3.3). The next sections investigate how these procedures handle the French sample problem.

4 Learning French Gender Allomorphy

In section 2, we showed through an example with three meanings and 20 constraints how an aspect of French liaison may be described in a multilevel parallel OT grammar. The present section illustrates how the evaluation and learning procedures of section 3, which were illustrated there with only one meaning and 10 constraints, work out for the complete problem of section 2. We investigate what parallel and serial analyses are possible, and which of these are found by computer-simulated learners.

4.1 The Constraints

Our first grammar contains the constraints of section 2, except the constraint against phonetic schwa (*[ə]). We summarize the whole set of 19 constraints here.

Lexical-semantic constraints (0 in CON). All such constraints are in the candidate generator GEN; that is, they are so strong that not even a connection is allowed that would violate them. For instance, the meaning ‘husband’ cannot connect to the morpheme ⟨*acteur*_M⟩ or ⟨*voiture*_F⟩, in this simplified world. The effect of this is equivalent to the effect of inviolable lexical-semantic constraints like *‘husband’⟨*acteur*_M⟩. The connections between ‘actor’ and ⟨*acteur*_M⟩ and between ‘good’ and ⟨*bon*⟩ therefore do not violate any constraints in CON.

Morphosyntactic constraints (1). Here we have only the gender disagreement constraint *⟨FM⟩, which is violated in ⟨*bon*-F; *acteur*_M⟩, ⟨*bon*-F; *mari*_M⟩, and ⟨*bon*-M; *voiture*_F⟩.

Lexical-phonological constraints (8). Many of these are in GEN, but the ones relevant to the allowed connections in figure 2 are *⟨*bon*-F⟩|bɔ̃|, *⟨*bon*-M⟩|bɔ̃|, *⟨*bon*-F⟩|bɔ̃|, *⟨*bon*-M⟩|bɔ̃|, *(M)|ə|, *(M)|∅|, *(F)|ə|, and *(F)|∅|. The last of these is violated for example in the connection from ⟨*bon*-F; *mari*_M⟩ to |bɔ̃n+∅#maʁi|. This constraint subset is formulated in an exhaustive way and handles allomorphy at the morpheme level—that is, whether feminine and masculine stems are allowed to be different in underlying form.

Faithfulness constraints (4). We have *|ɔ̃|/ɔ̃n/, *|ɔ̃n|/ɔ̃/, *|ə|// (against schwa deletion), and *|/ə| (against schwa insertion). Exhaustive connectivity (within CON) between underlying and surface form would also require antifaithfulness constraints such as *|ɔ̃|/ɔ̃/ and *|ɔ̃n|/ɔ̃n/, but these are not included in our example.

Structural constraints (2). We have the hiatus-avoiding constraint */V.V/, which is violated by the second syllable in /.bɔ̃.ak.tœʁ./, by the third syllable in /.bɔ̃.nə.ak.tœʁ./, and by any form

that starts with /bɔ̃.ə/ (i.e., /bɔ̃.ə.ak.tœʁ./ violates it twice). We also have the constraint */n./, which militates against /n/ in coda position.

Cue constraints (4). We have */ɔ̃/[ɔ̃n], */ɔ̃n/[ɔ̃], */ə/[], and *//[ə]. Exhaustive connectivity (within CON) between surface and phonetic form would also require “perverse” constraints such as */ɔ̃/[ɔ̃] and */ɔ̃n/[ɔ̃n], but these are not included in our example so as not to introduce too much arbitrariness at the phonology-phonetics interface.

4.2 Grammars That Work

For each of the three meanings there are 32 routes through the graph to arrive at the correct French phonetic form. When looking only at the graphs and ignoring constraints, we would therefore predict that there are $32^3 = 32,768$ possible analyses of the French data. The 19 constraints of section 4.1 severely restrict this number. A brute-force search⁷ shows that with this constraint set, there are only six different analyses that produce the correct phonetic forms for each of the three meanings. Three of these analyses are shown in figures 12–14: each of these figures shows a triplet of graphs that represents a grammar-cum-analysis of the French data (just as a set of tableaux does in two-level OT).

The analyses in figures 12–14 constitute two crucially parallel analyses and one “serial” analysis.

The first crucially parallel analysis (“PU”) is seen in figure 12. The morphemes ⟨bon-M⟩ map to the underlying form |bɔ̃n| if followed by a vowel in the surface form, but to the underlying form |bɔ̃| if followed by a consonant in the surface form. Hence, a “later” consideration in the phonology—namely, the constraint */V.V/ at the surface level—influences an “earlier” choice at the underlying level. Such bottom-up influences in production can occur only in a parallel evaluation, not in a serial evaluation.

The second crucially parallel analysis (“PG”) is seen in figure 13. The meaning ‘good’, applied to a masculine noun, maps to the morphemes ⟨bon-M⟩ if followed by a consonant in the surface form, but to the morphemes ⟨bon-F⟩ if followed by a vowel in the surface form. This is Encrevé-Lambert’s (1971) and Tranel’s (1996) gender allomorphy of section 2.2. Again, a “later” consideration influences an “earlier” choice, this time at the morphemic level. And again, this type of analysis can occur only in a parallel evaluation, not in a serial evaluation.

The “serial” analysis—that is, an analysis in our parallel model that could also occur in a serial model—is seen in figure 14. From the meaning level to the morpheme level, the contrast between masculine and feminine is maintained in that ‘good’ maps to ⟨bon-M⟩ before masculine nouns and to ⟨bon-F⟩ before feminine nouns. From the morphemic to the underlying level, the contrast between masculine and feminine is maintained in that the morphemes ⟨bon-M⟩ map to

⁷ Technically, this was done by applying Batch Constraint Demotion (Tesar and Smolensky 1993) on each of the 32,768 analyses and seeing whether the algorithm converged. Cycling through all possible analyses like this is feasible for the current toy example, but stops being feasible if the number of forms per level grows much larger.

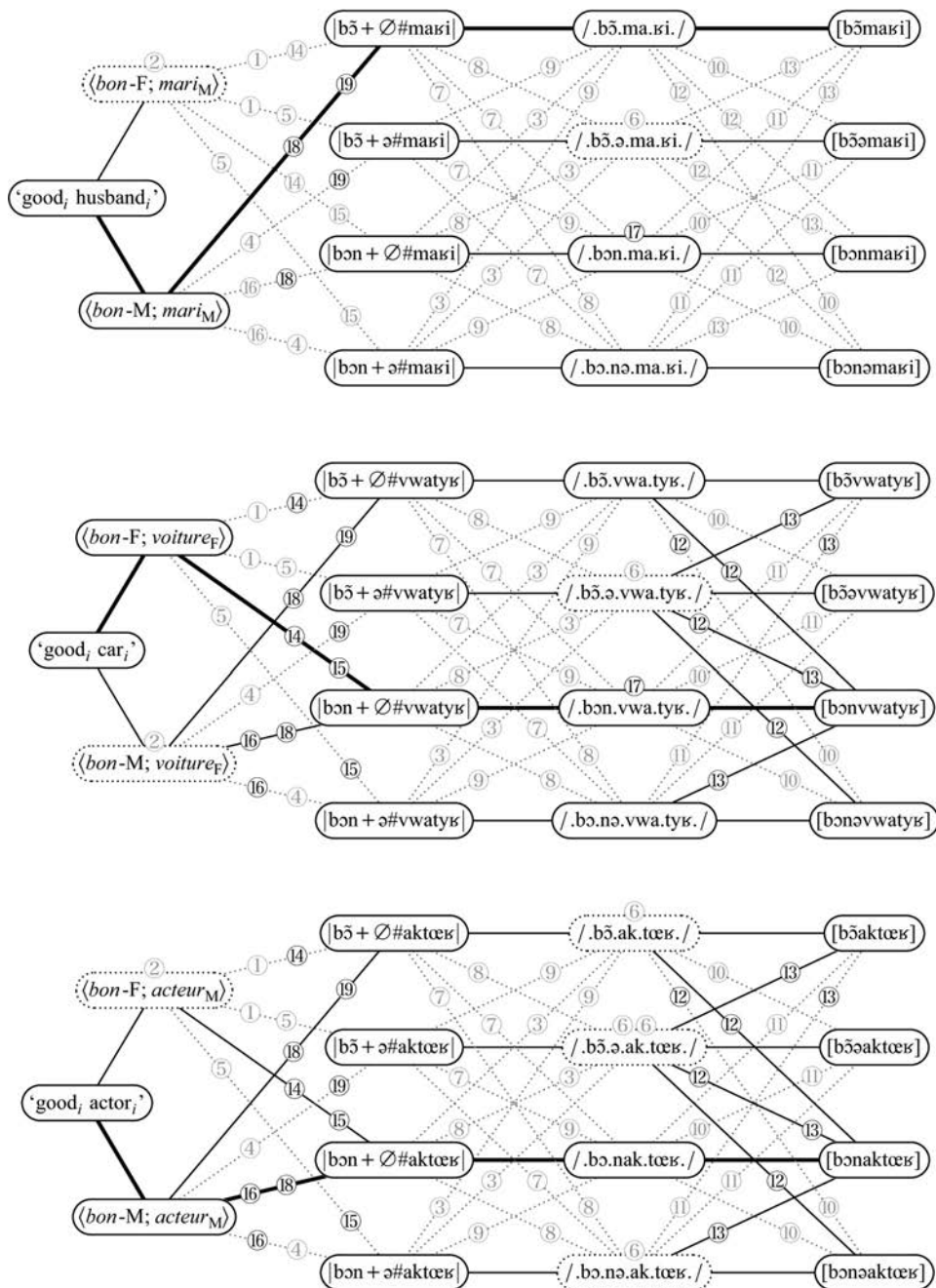


Figure 12

Parallel analysis with allomorphy in the underlying form (“PU”). Ranking: ① $*(bon-F)|b\delta| \gg$ ② $*(FM) \gg$ ③ $*|c\eta|/\delta/ \gg$ ④ $*(M)|c| \gg$ ⑤ $*(F)|c| \gg$ ⑥ $*/V.V/ \gg$ ⑦ $*|\delta|/c\eta/ \gg$ ⑧ $*||c| \gg$ ⑨ $*|c|// \gg$ ⑩ $*|/c| \gg$ ⑪ $*/c\eta|/\delta| \gg$ ⑫ $*/\delta|/c\eta| \gg$ ⑬ $*/c|/| \gg$ ⑭ $*(F)|\emptyset| \gg$ ⑮ $*(bon-F)|b\eta| \gg$ ⑯ $*(bon-M)|b\eta| \gg$ ⑰ $*/n./ \gg$ ⑱ $*(M)|\emptyset| \gg$ ⑲ $*(bon-M)|b\delta|$.

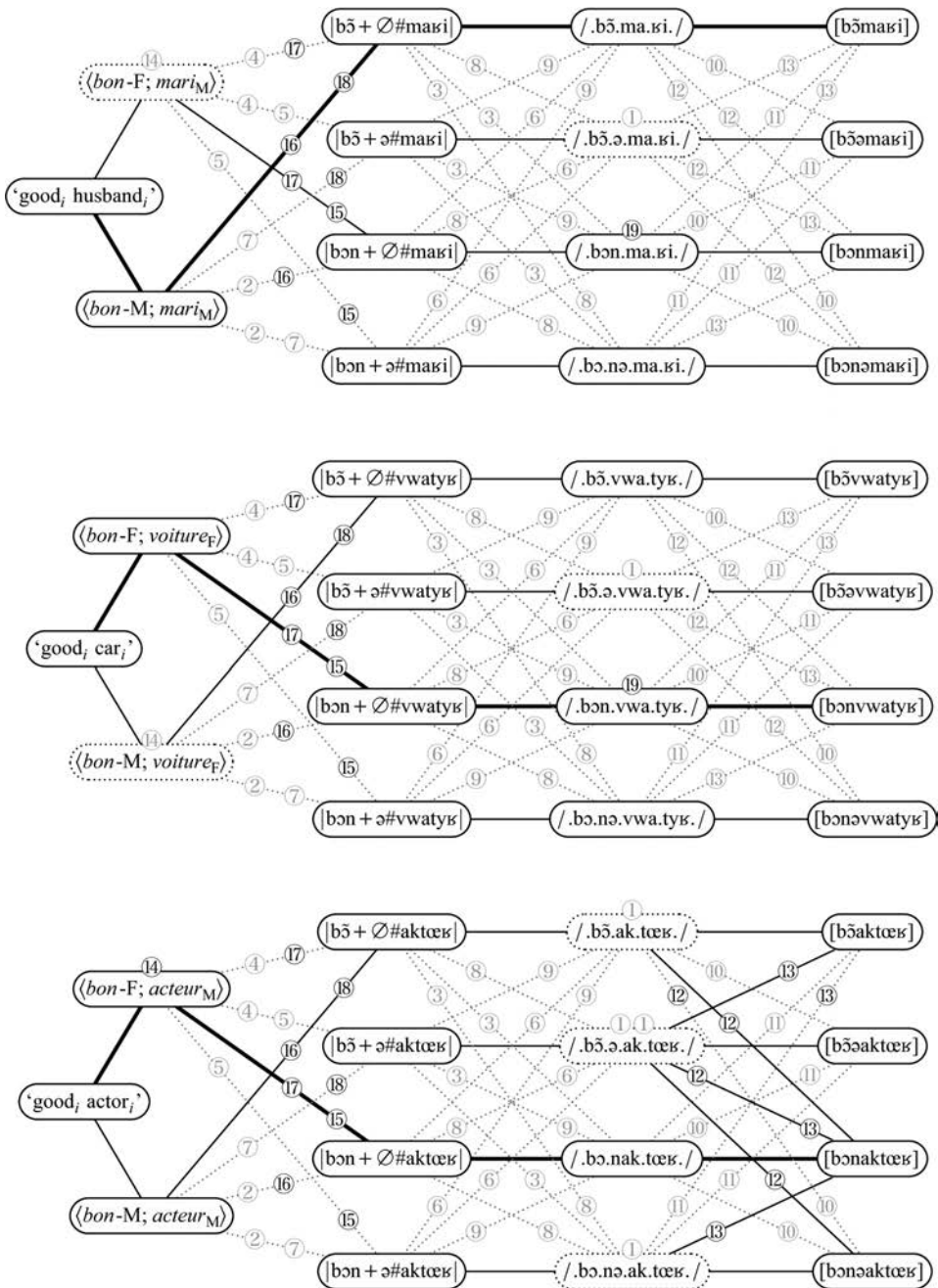


Figure 13

Parallel analysis with true gender allomorphy ('PG'). Ranking: ① $*/V.V/ \gg$ ② $*(\text{bon-M})|\text{b}\text{on}| \gg$ ③ $*/\delta|/\text{on}| \gg$ ④ $*(\text{bon-F})|\text{b}\delta| \gg$ ⑤ $*(F)|\text{a}| \gg$ ⑥ $*/\text{on}|/\delta/ \gg$ ⑦ $*(M)|\text{a}| \gg$ ⑧ $*/|\text{a}| \gg$ ⑨ $*/\text{a}|// \gg$ ⑩ $*/|/\text{a}| \gg$ ⑪ $*/\text{on}|[\delta] \gg$ ⑫ $*/\delta|[\text{on}] \gg$ ⑬ $*/\text{a}|/| \gg$ ⑭ $*(FM) \gg$ ⑮ $*(\text{bon-F})|\text{b}\text{on}| \gg$ ⑯ $*(M)|\emptyset| \gg$ ⑰ $*(F)|\emptyset| \gg$ ⑱ $*(\text{bon-M})|\text{b}\delta| \gg$ ⑲ $*/n./.$

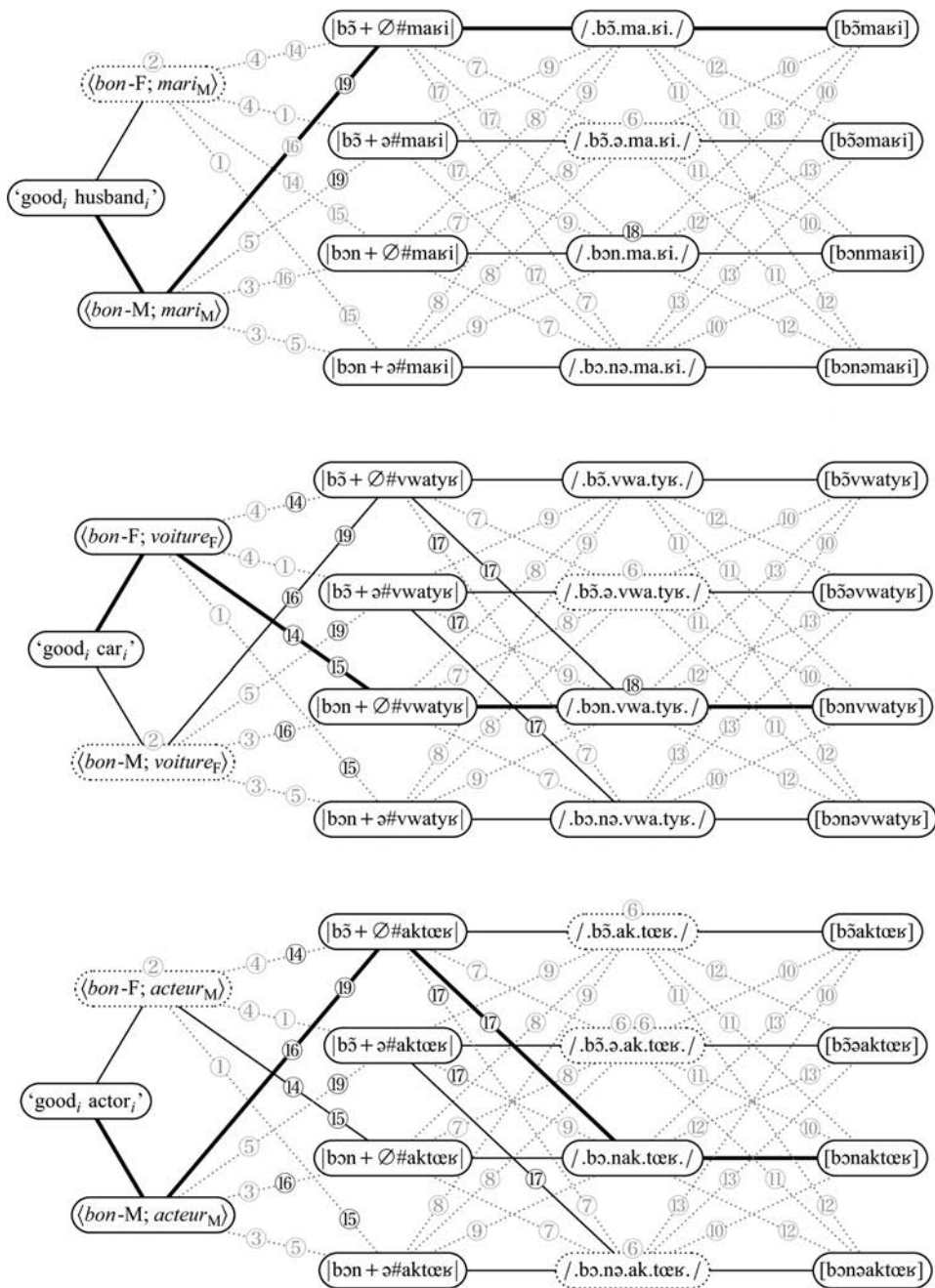


Figure 14

“Serial” analysis: phonological *n*-insertion (“SN”). Ranking: ① *⟨F⟩[ə] >> ② *⟨FM⟩ >> ③ *⟨bon-M⟩|bon| >> ④ *⟨bon-F⟩|bɔ̃| >> ⑤ *⟨M⟩[ə] >> ⑥ */V.V/ >> ⑦ */[ə]/ >> ⑧ *|ɔ̃n|/ɔ̃/ >> ⑨ *|ə|// >> ⑩ */[ə]/[] >> ⑪ */ɔ̃/[ɔ̃n] >> ⑫ *//[ə] >> ⑬ */[ə]/[ɔ̃] >> ⑭ *⟨F⟩|∅| >> ⑮ *⟨bon-F⟩|bɔ̃n| >> ⑯ *⟨M⟩|∅| >> ⑰ *|ɔ̃|/ɔ̃n/ >> ⑱ */n./ >> ⑲ *⟨bon-M⟩|bɔ̃|.

underlying $[b\bar{\sigma}]$ and the morphemes $\langle \text{bon-F} \rangle$ map to underlying $[b\bar{\sigma}n]$. The phonological constraint $*/V.V/$ enforces its influence only at the surface level, where it converts underlying $[b\bar{\sigma}]$ to surface $/b\bar{\sigma}n/$ if a vowel follows. This serial analysis (labeled “SN” as a reference to *n*-insertion) cannot handle data beyond the current toy example, because it cannot explain why $[b\bar{\sigma}]$ should become $/b\bar{\sigma}n/$ but $[m\bar{\sigma}]$ may become $/m\bar{\sigma}n/$ (Dell 1970, Selkirk 1972).

The remaining three analyses are similar to those of the graphs of figures 12–14, but have a schwa in some masculine intermediate forms. PU (figure 12) has a variant (“PU $\bar{\sigma}$ ”) with schwa in both masculine underlying forms: $[b\bar{\sigma}+\bar{\sigma}\#m\bar{\alpha}xi]$ and $[b\bar{\sigma}n+\bar{\sigma}\#akt\bar{\alpha}x\bar{\epsilon}]$.⁸ PG (figure 13) has a variant (“PG $\bar{\sigma}$ ”) with schwa in the consonantal masculine underlying form $[b\bar{\sigma}+\bar{\sigma}\#m\bar{\alpha}xi]$ only.⁹ Finally, SN has a variant (“SN $\bar{\sigma}$ ”) with schwa in both masculine underlying forms, which are now $[b\bar{\sigma}+\bar{\sigma}\#m\bar{\alpha}xi]$ and $[b\bar{\sigma}+\bar{\sigma}\#akt\bar{\alpha}x\bar{\epsilon}]$.¹⁰ One can say that for all analyses it is immaterial whether the underlying form has $[\emptyset]$ or $[\bar{\sigma}]$; the ranking of $*(M)|\emptyset|$ with respect to $*(M)|\bar{\sigma}|$ and $*|\bar{\sigma}||$ determines the masculine underlying ending, but this ranking does not influence anything else in the analysis. The reason why no schwas occur in the feminine underlying forms is that $*/V.V/$ is capable of “deleting” underlying schwas from $[b\bar{\sigma}+\bar{\sigma}\#m\bar{\alpha}xi]$, $[b\bar{\sigma}+\bar{\sigma}\#akt\bar{\alpha}x\bar{\epsilon}]$, and $[b\bar{\sigma}n+\bar{\sigma}\#akt\bar{\alpha}x\bar{\epsilon}]$, but not from $[b\bar{\sigma}n+\bar{\sigma}\#v\bar{w}at\bar{y}\bar{\epsilon}]$ (but see section 4.6).

The serial analysis of section 2.1 does not appear here, because $*[\bar{\sigma}]$ has not been included in the constraint set yet. See section 4.6 (and figure 15) for analyses that become possible when the constraint set is larger.

4.3 Error-Driven Learning

We have shown that the set of 19 constraints of section 4.1 is compatible with six distinct analyses that produce the correct phonetic form for each meaning. Some of those analyses had been proposed before, but some were novel. It remains to be investigated which (if any) of these six analyses are *learnable*. After all, it is possible that there exist analyses that are allowed by factorial typology (i.e., representable by a constraint ranking) but for which no learning path exists. It then becomes interesting to see which of the six analyses *virtual* (i.e., computer-generated) learners will find: starting with a certain initial state and following a certain reranking algorithm, will they come up with a serial analysis known from the literature, such as the one in section 2.1, or with a parallel analysis known from the literature, such as the one in section 2.2, or with one of the novel analyses?

To assess learnability, we supply a *virtual learner* with 10,000 pairs of meaning and phonetic form randomly drawn from the French data in section 3.3; that is, each of the three form-meaning pairs will occur approximately 3,300 times in the learner’s input. During this process, the learner

⁸ A possible ranking is $*(F)|\bar{\sigma}| \gg *(FM) \gg */V.V/ \gg *|\bar{\sigma}|/\bar{\sigma}n/ \gg *(bon-F)|b\bar{\sigma}| \gg *(M)|\emptyset| \gg *||\bar{\sigma}/ \gg *|\bar{\sigma}n|/\bar{\sigma}/ \gg */\bar{\sigma}|/\bar{\sigma}n| \gg */\bar{\sigma}n|/\bar{\sigma}| \gg */\bar{\sigma}/[] \gg *||\bar{\sigma}| \gg *(bon-M)|b\bar{\sigma}n| \gg *(bon-F)|b\bar{\sigma}n| \gg *(F)|\emptyset| \gg *(M)|\bar{\sigma}| \gg */n./ \gg *|\bar{\sigma}|| \gg *(bon-M)|b\bar{\sigma}|$.

⁹ A possible ranking is $*(F)|\bar{\sigma}| \gg *(M)|\emptyset| \gg *(bon-M)|b\bar{\sigma}n| \gg */V.V./ \gg *||\bar{\sigma}/ \gg *(bon-F)|b\bar{\sigma}| \gg *|\bar{\sigma}|/\bar{\sigma}n/ \gg *|\bar{\sigma}n|/\bar{\sigma}/ \gg */\bar{\sigma}|/\bar{\sigma}n| \gg */\bar{\sigma}/[] \gg *||\bar{\sigma}| \gg */\bar{\sigma}n|/\bar{\sigma}| \gg *(FM) \gg */n./ \gg *(bon-M)|b\bar{\sigma}| \gg *(M)|\bar{\sigma}| \gg *|\bar{\sigma}|| \gg *(bon-F)|b\bar{\sigma}n| \gg *(F)|\emptyset|$.

¹⁰ A possible ranking is $*(M)|\emptyset| \gg *(bon-M)|b\bar{\sigma}n| \gg *(bon-F)|b\bar{\sigma}| \gg *(FM) \gg */V.V./ \gg *(F)|\bar{\sigma}| \gg *||\bar{\sigma}/ \gg *|\bar{\sigma}n|/\bar{\sigma}/ \gg */\bar{\sigma}n|/\bar{\sigma}| \gg */\bar{\sigma}/[] \gg */\bar{\sigma}|/\bar{\sigma}n| \gg *||\bar{\sigma}| \gg *(bon-M)|b\bar{\sigma}| \gg *(F)|\emptyset| \gg *|\bar{\sigma}|/\bar{\sigma}n/ \gg *(bon-F)|b\bar{\sigma}n| \gg *(M)|\bar{\sigma}| \gg */n./ \gg *|\bar{\sigma}||$.

is equipped only with a constraint set (the 19 constraints), a candidate generator (the graphs of figure 2), a single current grammar hypothesis (i.e., a current constraint ranking), and at most a single datum (form-meaning pair) that she is currently handling. On each datum, the learner performs a *virtual production*; that is, she computes the optimal path from the given meaning according to the evaluation procedure of section 3.2 and compares this with the path computed by robust interpretive parsing (section 3.3). If the paths are different, the learner takes action by changing her constraint ranking (this is therefore *error-driven learning*). After 10,000 data, we stop the learner and check whether her final constraint ranking is correct—that is, whether the ranking maps the three French meanings to the correct French phonetic forms. Note that the learning procedure works *online*: the learner maintains a single ranking at each time and considers a single data pair at a time, without any memory of previous hypotheses or previous data.

4.4 The Random Baseline Learner

It has been argued that learning algorithms should be checked against a *random baseline* (for parameter setting: Berwick and Niyogi 1996; for OT: Jarosz 2013b). In our case, the random baseline learner starts by randomly choosing a constraint ranking out of the 19! possible rankings, and when her ranking fails on a datum, she randomly chooses a new constraint ranking. When we simulated 1,000 learners in this way, they all turned out to have a correct French grammar after 10,000 data. Counts of their resulting analyses are in the top row of table 1. We conclude that the learners have a preference for the “serial” analyses (58%) over the parallel analyses (42%), and for the schwa-less analyses (78%) over the analyses that posit a schwa somewhere along the route (22%). Since in this algorithm a grammar no longer changes after it is correct, these preferences are easy to explain: they correspond to the number of rankings (out of the 19! possible ones) that yield each analysis.

The fact that all 1,000 learners succeeded can be explained by the probability of guessing a correct ranking by chance. When we drew 1,000,000 random rankings of the 19 constraints, about 0.62 percent of these rankings represented a correct grammar for the French data. This means that a random baseline learner will run into a correct grammar after making on average 160 errors, a number that is reached after at most approximately 480 pieces of data (the worst case, which occurs if all errors are made for only one of the three data pairs). The probability that a learner has not encountered a correct grammar after 10,000 data, if the incidence of such grammars is 1/480, is approximately 10^{-9} .

Table 1

Proportions in which the six possible analyses are found by two kinds of learners. For random baseline learners, see section 4.4; for “weighted uncanceled” learners, see section 4.5.

Analysis	PU	PG	SN	PU \emptyset	PG \emptyset	SN \emptyset	SN~PU	SN~SN \emptyset	Total
Proportion of random baseline learners	25%	7%	46%	8%	2%	12%	0	0	100%
Proportion of “weighted uncanceled” learners	1.7%	0?	21%	0.01%	0?	0.9%	0.16%	0.04%	24%

The problem with the random baseline, however, is that it does not scale. Berwick and Niyogi (1996) took Gibson and Wexler's (1994) grammar of 3 parameters and showed that a random error-driven selection out of the 2^3 (= 8) possible grammars results in more (namely, 100%) successes and faster convergence than Gibson and Wexler's 'local and greedy' algorithm, which takes on the order of 3^2 (= 9) error steps; the problem here is that Berwick and Niyogi's baseline would probably fail if the number of parameters were 30 instead of 3, because 2^{30} (= 1,073,741,824) is much more than 30^2 (= 900). Likewise, the random baseline in the OT case tends to scale exponentially with the number of constraints: for instance, the word-learning model by Hamann, Apoussidou, and Boersma (2011), when applied to 50 words, would have only $50! \times 50!$ correct rankings of lexical constraints out of $100!$ possible rankings, given a chance of finding a correct grammar only once in every 10^{29} guesses, so there is little hope that those 100 constraints will be ranked correctly by random selection within any learner's lifetime. We expect that a simulation involving more liaison forms and lexical constraints than the toy French language of this article will likewise not be feasible with a random baseline learner.

In the following section, we therefore try out the 'local and greedy' reranking procedure of section 3.3, and several variants of it.

4.5 Incremental Learning Procedures

The opposite of a random reselection procedure is an incremental procedure: in case the current grammar hypothesis fails on the incoming datum, only a few constraints are reranked, and the remainder retain their current ranking. One example of this is the reranking procedure in section 3.3, which is the standard version of the Gradual Learning Algorithm (Boersma and Hayes 2001). This assumes *Stochastic OT*, in which all constraints have numerical ranking values, and indeed the initial state of the learner in our simulations is that all constraints are ranked at the same height of 100.0. Evaluation proceeds by temporarily adding to each constraint's ranking a random value drawn from a Gaussian distribution with standard deviation 2.0 (the *evaluation noise*). When a learning pair comes in, the learner uses the evaluation noise to determine a constraint ranking, and then uses this ranking to compute both her virtual production and her robust interpretive parsing ('correct' path). If these two paths differ, all constraints that prefer (i.e., have fewer violations in) the produced path (\otimes) are demoted by a value of 1.0 (the *plasticity*), and all constraints that prefer the 'correct' path (\checkmark) are promoted by 1.0. The effect is that the grammar moves closer to a ranking where the supposedly correct candidate may win.

When we simulate 100 learners with this 'symmetric all' reranking procedure, however, none of them finds a correct grammar of French after 10,000 data. The learners typically get stuck in various types of limit cycles, alternating between a grammar that handles 'good actor' and 'good car' correctly and a grammar that handles 'good actor' and 'good husband' correctly. Figures 15 and 16 illustrate a type of limit cycle that is known from the literature (Tesar and Smolensky 2000:67, Boersma and Pater 2016:423). The constraint ranking in figure 15 correctly produces 'good husband' as [bõmasi] (not shown), but incorrectly produces 'good car' as [bõvwyatʁ], as shown by the semithick path and the pointing finger. When the correct phonetic form [bõnvwyatʁ] comes in, leading to the 'correct' thick path, a comparison between the two paths will lead to a demotion of constraints ⑩ *|ʃ|/ɔn/ and ⑪ */ə/[] and a promotion of constraint

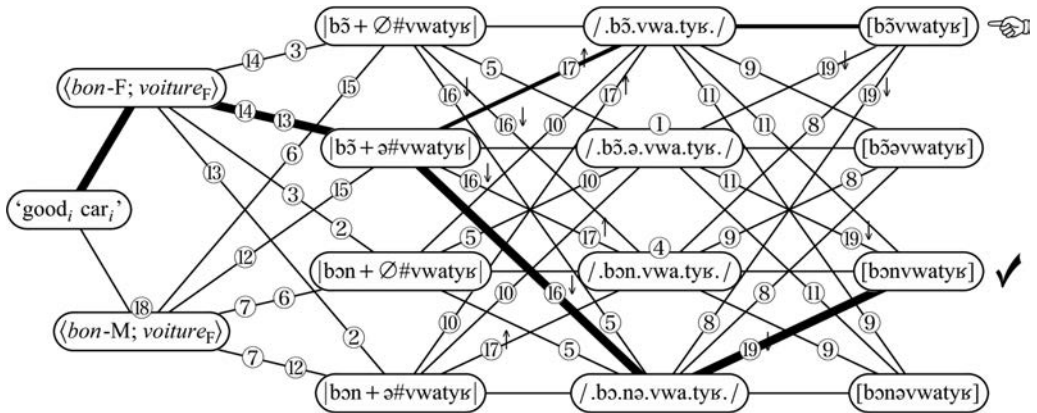


Figure 15

A ranking appropriate for [b̄ɔ̄maksi] but not for [b̄ɔ̄nvwatȳκ]: upon encountering [b̄ɔ̄nvwatȳκ], the learner will promote constraint 17 and demote constraints 16 and 19.

17 *|ə|/. The combination of movements of 16 *|ɔ̄|/ɔ̄n/ and 17 *|ə|/ will typically result in these constraints becoming ranked in the opposite order. This change is shown in figure 16, where *|ə|/ is now labeled 16 and *|ɔ̄|/ɔ̄n/ is labeled 17. This new ranking, however, now causes the grammar to fail on the input ‘good husband’, producing [b̄ɔ̄nəmasi] as shown in the figure. When the correct phonetic form [b̄ɔ̄maksi] now comes in, the learning algorithm will demote 16 *|ə|/ and promote 17 *|ɔ̄|/ɔ̄n/. This movement will restore the original ranking of *|ɔ̄|/ɔ̄n/ over *|ə|/, leading again to a grammar that handles ‘good husband’ correctly but fails on ‘good car’.

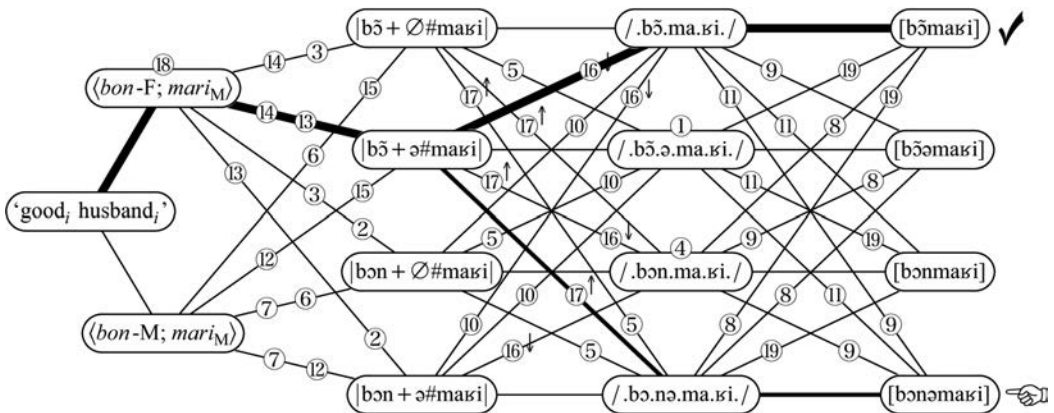


Figure 16

The ranking that results from the constraint movements in figure 15, which will now incorrectly produce [b̄ɔ̄nəmasi]. When confronted with [b̄ɔ̄maksi], the learner promotes one of the constraints demoted in figure 15 and demotes the constraint that was promoted in figure 15.

We can identify two problems with the events in figures 15 and 16. The first problem is that $*|\mathfrak{z}|/\mathfrak{on}/$ and $*|\mathfrak{a}||$ will swap places eternally, never allowing the grammar to leave this cycle. The second problem is that the net result of the figure 15 – figure 16 cycle is that constraint ⑨ $*/\mathfrak{a}/[]$, which was already bottom-ranked, has moved further down the hierarchy; in a situation of stochastic ranking, this fruitless eternal downward “pumping” reduces the possibility that $*/\mathfrak{a}/[]$ will ever play a role again. We will now discuss various attempts to improve on this problem.

The pumping effect can be reduced by a variant reranking procedure, namely, “weighted uncanceled” (Apoussidou 2007:174). This update rule is similar to “symmetric all,” except that the value subtracted from the ranking of \mathfrak{E} -preferring constraints is now equal to the plasticity divided by the number of these constraints, and the value added to the ranking of \checkmark -preferring constraints equals the plasticity divided by the number of these constraints (if either of these numbers of constraints is zero, no constraint moves). In figure 15, for instance, constraints ⑥ and ⑨ are demoted by only 1/2 each and constraint ⑰ is promoted by 1, and the combination of figures 15 and 16 leads to ⑰ staying fixed, ⑥ rising by 1/2, and ⑨ falling by 1/2. In general, this scheme leads to less downward pumping than “symmetric all,” because the amount by which constraints can fall is kept within bounds by the amount that other constraints can rise, and this is in turn bounded by the other constraints in the grammar: in our example, for instance, constraint ⑩ $*|\mathfrak{z}|/\mathfrak{on}/$ can never reach beyond the top of the hierarchy, because if this constraint becomes high-ranked, the learner will choose as the “correct” path a path that does not violate it. The result is that both ⑥ and ⑨ get more chances to interact with other constraints under the “weighted uncanceled” scheme than under the “symmetric all” scheme, and this is a general difference between the schemes that is not specific to the toy case at hand.

For our French example, it is indeed the case that the “weighted uncanceled” reranking procedure works better than “symmetric all”: about 24% of 10,000 virtual learners succeeded. The analyses that the succeeding learners came up with are summarized in the bottom row of table 1. Most found the “serial” analysis SN, 90 found SN \mathfrak{a} , and 170 found PU (same preference as the random baseline learners). A few (16) learners came up with a variable grammar in which $*\langle \text{bon-M} \rangle | \text{b}\mathfrak{on} |$ and $*|\mathfrak{z}|/\mathfrak{on}/$ were ranked at the same height, leading to a 50–50 variation between the SN and PU analyses; in this example, we see that the ranking difference between a “serial” and a crucially parallel grammar can be minimal (the learner cannot notice this; to a learner, all grammars are parallel and the potential existence of a serial equivalent is inaccessible). Finally, 4 of the 10,000 learners came up with a variable grammar in which $*\langle \text{M} \rangle | \mathfrak{a} |$ and $*\langle \text{M} \rangle | \emptyset |$ were ranked at the same height, leading to a 50–50 variation between the SN and SN \mathfrak{a} analyses. We see that only three of the six analyses outlined in section 4.2 emerge with any frequency in the OT grammars, and that variable grammars that never occur for random baseline learners (for whom two rankings have probability zero of being equal) are also possible.

A point of concern in figure 15 is the demotion of ⑨: demoting constraints that are ranked below the highest-ranked \checkmark -preferring constraint (⑰) cannot really help to improve the decision. Tesar and Smolensky’s (1998) Error-Driven Constraint Demotion (EDCD) therefore prevents the demotion of ⑨: in this scheme, all \mathfrak{E} -preferring constraints ranked above the highest-ranked

✓-preferring constraint are demoted to a value just below this highest-ranked ✓-preferring constraint, and no constraint is promoted. In figure 15, ⑩ $*|\delta|/\text{on}/$ will be demoted below ⑦ $*|\partial|//$, and in figure 16, $*|\partial|//$ will be demoted below $*|\delta|/\text{on}/$. These two constraints will continue to tumble down in this way, and once they pass constraint ⑨ $*|\partial|/[]$, they will drag it along down the hierarchy. The chances for these three constraints to interact with the rest of the hierarchy will diminish even faster than with “symmetric all,” and indeed in our simulations with EDCD (with stochastic ranking; Boersma 2009), none of the 10,000 simulated learners succeeded in correctly learning a ranking for the three French pieces of data.

Magri’s (2012) update rule limits some of the demotions of EDCD, while retaining the advantage of not demoting ⑨ in figure 15: in this scheme, all ✖-preferring constraints ranked above the highest-ranked ✓-preferring constraint are demoted by 1.0, whereas all constraints that prefer the “correct” path are promoted by 1.0 multiplied by {the number of constraints being demoted} and divided by {the number of constraints being promoted plus one}. In figure 15, this means that ⑩ falls by 1 and ⑦ rises by 1/2, and in the combination of figures 15 and 16, both constraints end up being demoted by 1/2, while ⑨ is pumped down once the other two constraints have reached it. Regarding this behavior, this scheme can be expected to work better than EDCD but not better than “weighted uncanceled,” and indeed none of 10,000 simulated learners (with stochastic ranking) turned out to be able to learn from the data.

The relative success of the various update rules corroborates earlier comparisons in the literature, where “symmetric all” had more success than EDCD (Boersma 2003) and “weighted uncanceled” had more success than “symmetric all” (Apoussidou 2007).

A general strategy for improving solutions to difficult optimization problems is to add randomness (Kirkpatrick, Gelatt, and Vecchi 1983), and this indeed turns out to help in the current case. We performed an additional series of simulations for the 19-constraint grammar of section 4.1 under the Generalized Robust Interpretive Parsing (GRIP) strategy introduced by Biró (2013). Recall that under regular RIP, the sets of constraints eligible for promotion and demotion are decided by comparing an (incorrectly) optimal candidate with a parsed target candidate containing a target form. Under GRIP, the optimal candidate is instead compared with a Boltzmann-weighted mean of the entire set of candidates containing the target form. Biró (2013) argues that by maintaining multiple hypotheses over the correct parse in this manner, the learner is more likely to converge on a grammar consistent with all data. The settings for simulations performed with GRIP were identical to those reported above, except with regard to evaluation noise: as Tamás Biró (pers. comm.) suggests, the repeated shuffling of the hierarchy through evaluation noise may not be compatible with GRIP’s notion of a decreasing “temperature vector.” Indeed, we found that no GRIP learners converged on a correct French grammar within 40,000 data under the standard evaluation noise of 2.0. On the other hand, when the evaluation noise was set to a very small value of 10^{-9} , GRIP turned out to outperform regular RIP for our virtual learners of French: 100 out of 100 “weighted uncanceled” learners, 84 out of 100 “symmetric all” learners, 33 out of 100 learners using Magri’s (2012) update rule, and 0 out of 100 EDCD learners converged to a correct grammar under GRIP. Again, we see the usual success relations between the algorithms.

As stated in the beginning of this section, the virtual learners in our simulations find optimal candidates through a hierarchy that is stochastically influenced by evaluation noise, and the same draw of the evaluation noise is used for virtual production and for interpretive parsing. A study by Jarosz (2013a) suggests, however, that virtual production and interpretive parsing are performed with different draws of the evaluation noise. Jarosz reports a case in Stochastic OT where this ‘‘resampling’’ improves learning. When repeating our simulations described in the beginning of this section with resampling (and reranking the constraints only if the *phonetic form* differed between the two paths), we found that for all update rules the chances of a learner settling on a correct grammar improved: 83 out of 100 ‘‘weighted uncanceled’’ learners, 39 out of 100 ‘‘symmetric all’’ learners, and 45 out of 100 learners using Magri’s (2012) update rule succeeded in finding a correct French grammar. The increase in learning success reported by Jarosz thus seems to extend to multilevel grammars.

4.6 *A Grammar with a Constraint against Phonetic Schwa*

In all, the success of the learners with the 19 constraints of section 4.1 is modest. A possible cause is that the number of constraints is too low. We therefore tried the constraint set of section 2.1, which contains the same 19 constraints as the grammar of section 4.1, plus an articulatory constraint *[ə] that is violated by every occurrence of a schwa in the phonetic form (the constraint seen in the serial account of tableau (12)). Again, only ‘‘weighted uncanceled’’ learners succeeded, in this case 80 out of 100. The analysis found most frequently by the learners (32 times) was one not possible in the original 19-constraint grammar: a variant of SN (figure 14) in which the female form contains a schwa on the underlying and surface levels, which is deleted in the phonetic form.

The traditional serial of analysis of section 2.1 was found by 1 of the 80 learners, and it is shown in figure 17. The ranking of figure 17 is of course not the only ranking that yields these paths. A shift of the four relevant constraint classes such that the rankings *between* these classes become {lexical-semantic, morphosyntactic} >> lexical-phonological >> {faithfulness, structural} >> {cue constraints, *[schwa]} leads to the same winning paths as long as the rankings *within* the classes are preserved; precisely this property is what allows us to call the analysis serial.

A fairly large group of learners found a crucially parallel analysis in which a feminine underlying schwa was deleted from the surface form by the ranking of the lower-level constraints *[ə] and */ə/[] over the higher-level constraints */ə// and */n./.

4.7 *Harmonic Grammar*

The algorithm in section 3.2 is specific to the constraint-ranking decision mechanism of OT. It has been suggested that learners that instead use the decision mechanisms of Harmonic Grammar (HG) or one of its variants might perform better on multilevel problems than OT learners (Boersma and Pater 2016). HG grammars employ weighted constraints instead of the ranked constraints of OT. In our graph-based representation, the evaluation procedure reduces in HG to finding the

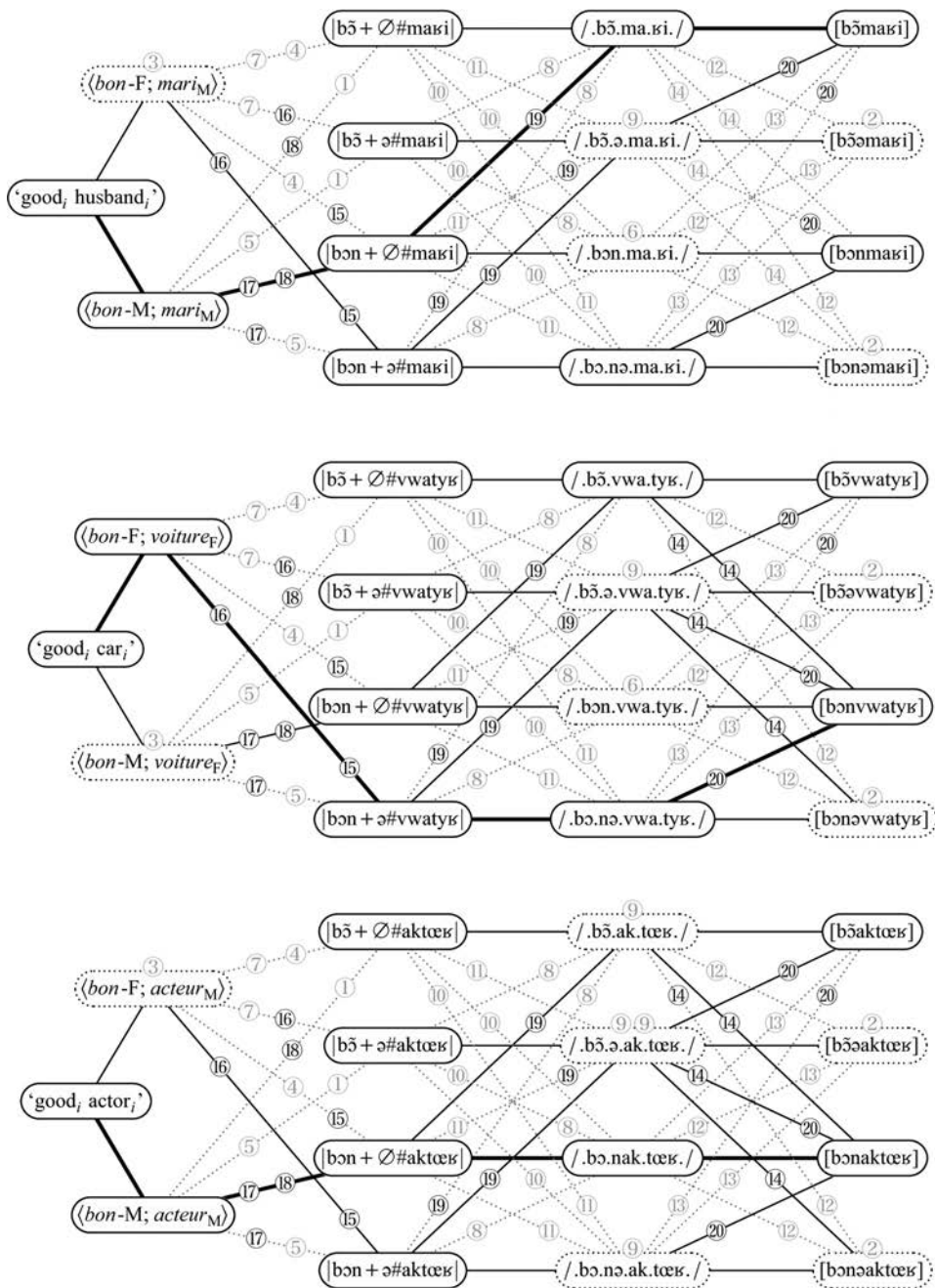


Figure 17

Serial analysis with phonetic schwa deletion. Ranking: ① *⟨bon-M⟩|b5| ≫ ② *[ə] ≫ ③ *(FM) ≫ ④ *(F)|∅| ≫ ⑤ *(M)|ə| ≫ ⑥ */n./ ≫ ⑦ *(bon-F)|b5| ≫ ⑧ *|ə|/| ≫ ⑨ */V.V/ ≫ ⑩ */5|/ə| ≫ ⑪ */|/ə| ≫ ⑫ */|[ə] ≫ ⑬ */ə|/[5] ≫ ⑭ */5|/ə| ≫ ⑮ *(bon-F)|bən| ≫ ⑯ *(F)|ə| ≫ ⑰ *(bon-M)|bən| ≫ ⑱ *(M)|∅| ≫ ⑲ *|ə|/[5]/ ≫ ⑳ */ə|/[].

shortest path, which is efficiently done with dynamic programming, as in the case of OT (section 3.3).

We carried out a number of simulations in which candidates were evaluated using HG (Legendre, Miyata, and Smolensky 1990) and its variants Maximum Entropy (Goldwater and Johnson 2003), Exponential HG (Boersma and Pater 2016), Positive HG (Boersma and Pater 2016), and Linear OT (Keller 2000). All of these were tested with two of the update rules discussed before, namely, “symmetric all” and “weighted uncanceled” (the other update rules we used above are specific to OT). The result was that for all of these decision mechanisms (except in some cases Exponential HG), the “weighted uncanceled” update rule was successful for all 100 learners (both with the 19-constraint set and with the 20-constraint set), and the “symmetric all” update rule was successful for 99 learners (with 20 constraints), 78 learners (with 19 constraints in HG and Maximum Entropy), or 50 learners (with 19 constraints in Positive HG and Linear OT). Indeed, HG-style learners performed better in the simulations than OT learners (replicating a tendency also reported in Boersma and Pater 2016), and, again, the “weighted uncanceled” update rule performed better than “symmetric all” and learners with an additional constraint against phonetic schwa again outperformed learners who had only the original 19 constraints.

Typologically speaking, HG tends to come with a larger set of possible languages than OT. While for the 19-constraint case only six analyses were possible using an OT constraint hierarchy with strict domination, the HG learners found several more weighted constraint rankings: besides five of the possible OT analyses of section 4.2 (the three analyses that the OT learners found in section 4.2, plus the one of figure 9, plus the true gender allomorphy analysis of figure 7), the learners found three analyses that rely on the possibility of having negative constraint weights. An example of an additional triplet of optimal candidates found by a small percentage of HG learners is the following:

‘good_i car_i’ ⟨*bon-F*; *voiture_F*⟩ |b̥+ə#vwaɥɛ| / .bɔ.nə.vwa.tɥɛ./ [bɔnvwaɥɛ]
 ‘good_i actor_i’ ⟨*bon-M*; *acteur_M*⟩ |b̥+∅#aktœɛ| / .bɔ.nak.tœɛ./ [bɔnakœɛ]
 ‘good_i husband_i’ ⟨*bon-M*; *mari_M*⟩ |b̥+∅#makɪ| / .b̥.ma.kɪ./ [b̥makɪ]

This analysis combines phonological *n*-insertion with schwa deletion at the phonetic level, and is not similar to any analysis previously proposed in the literature; the cause is, for example, that the path between / .bɔ.nə.vwa.tɥɛ./ and [bɔnvwaɥɛ] requires a negative weight for the constraint */ə/[], something that is impossible in OT. As the typological consequences of negative constraint weights are probably undesirable in general (Pater 2009), it is worthwhile to look at the analyses found by HG learners whose constraint weights are restricted to being positive. The analyses found by learners with such decision mechanisms (Exponential HG, Positive HG, Linear OT) all fell within the set of six OT-compatible analyses of section 4.2, which supports Pater’s (2009, 2016) claim that the typology of positive versions of HG is not so different from the typology of OT. In the end, while learnability criteria seem to favor HG over OT, as suggested both by Boersma and Pater (2016) and by the simulations in this article (as measured by the success rates of the learners), the choice between HG and OT has to be determined also by which of the two produces the better typologies.

4.8 Conclusion

In the OT framework, the “symmetric all,” “Magri,” and EDCD learners succeeded with neither constraint set, whereas the “weighted uncanceled” update rule was moderately successful. Whether these differences between the update rules for the present case reflect genuine differences in quality between the update rules or whether they are peculiar to the present case cannot yet be determined; a whole-language simulation performed on a large corpus of French data may shed light on this question.

Such a larger simulation is also needed if we want to find out whether a serial analysis or a parallel analysis is more appropriate for French. For instance, the serial analysis regards the *bon ~ bonne* alternation as phonological, whereas the parallel analysis regards this alternation as suppletive. Including in the simulation some data that are uncontroversially suppletive, such as the *ma ~ mon* or the *vieux ~ vieil(le)* alternation, may shift the preference of the virtual learners in the direction of a parallel analysis, whereas including more data that are possibly regarded as phonological, such as the alternation between the nouns *chien* [ʃjɛ̃] ‘dog’ and *chienne* [ʃjen] ‘bitch’, may shift the preference in the direction of a serial analysis. Such data, and more realistic and extensive data on schwa drop and schwa insertion, pose an interesting subject of future research. A specific expectation is that the analyses with underlying masculine schwas found in the present limited example will vanish.

5 The Relation to Complexity Reductions for Other Parameters Than Number of Levels

In this article, we constructed an algorithm for parallel evaluation across multiple levels of representations whose complexity is linear in the number of levels, while the size of the candidate set is exponential in the number of levels. As mentioned in section 1, this stands in a tradition of reducing exponential candidate sets to linear by graph-theoretic means: linearity has been achieved for the number of segments in the input and the number of autosegmental tiers (Ellison 1994, Eisner 1997, Riggle 2004). There is a difference in the kind of grammar and the kind of candidate generator between our work and this earlier work: when describing the relation between adjacent levels, we worked with small lists of candidates (enumerated in tableaux), just as in Prince and Smolensky’s (1993) formalization and in most practical work in OT, whereas Ellison, Eisner, and Riggle achieved their linearity results under the restriction that candidate sets can be represented as regular expressions, the “finite-state” assumption.

A super-efficient comprehensive model of evaluation in parallel multilevel OT would preferably be subexponential in the number of segments in the input *and* in the number of autosegmental tiers *and* in the number of levels of representation. Can this be achieved?

The finite-state models by Riggle (2004) achieve the evaluation of an infinite candidate set. One would like to apply that method to multilevel evaluation. However, finite-state transducers have a single input alphabet and a single output alphabet, whereas our French example works with at least four different alphabets (the underlying form and the surface form may both be written in the same phonological alphabet, but the other three levels are incommensurable with the phonological levels and with each other), and once one includes syntactic and semantic repre-

sentations, the number of alphabets will increase again. One could represent the French case as a concatenation of four finite-state transducers, but such an apparatus would perform only *serial* multilevel evaluation. For parallel evaluation, one would need a single giant transducer, with meaning as the input and phonetic form as the output; the alphabets of the intermediate levels would then remain unexpressed. Unfortunately, the size of the transducer seems to have to become impractically large. According to Riggle (2004:100), the number of states in an OT transducer is exponential in the number of constraints if the constraints work on different kinds of structures. In a multilevel case, constraints that work at different levels of representations do not share structures, so that the number of states seems to have to be exponential in the number of levels, which is exactly what the present account wants to avoid; for our 19-constraint case, the states probably number many thousands. Future work by finite-state and/or multilevel theorists may find a solution to this problem.

6 Conclusion

In this article, we illustrated that a parallel multilevel constraint grammar can be represented as a graph with a number of connections that is linear in the number of levels, although the number of candidate paths is exponential in the number of levels. We illustrated how this leads to efficient evaluation procedures and learning mechanisms whose computation times are also linear in the number of levels. Although for the time being we have to stay agnostic about whether our French example is best described with a serial or with a parallel grammar, the linear computation time helps to make parallel multilevel evaluation and learning feasible as a method of modeling phonological processing and acquisition. This kind of linearity may well become essential when we scale up to more realistic problems—for example, when we apply parallel multilevel evaluation in whole-language simulations.

References

- Anttila, Arto. 1997. Deriving variation from grammar. In *Variation, change and phonological theory*, ed. by Frans Hinskens, Roeland van Hout, and Leo Wetzels, 35–68. Amsterdam: John Benjamins.
- Apoussidou, Diana. 2007. The learnability of metrical phonology. Doctoral dissertation, University of Amsterdam.
- Apoussidou, Diana, and Paul Boersma. 2004. Comparing two Optimality-Theoretic learning algorithms for Latin stress. In *WCCFL 23: Proceedings of the 23rd West Coast Conference on Formal Linguistics*, ed. by Vineeta Chand, Ann Kelleher, Angelo J. Rodríguez, and Benjamin Schmeiser, 29–42. Somerville, MA: Cascadilla Press.
- Bellman, Richard. 1957. *Dynamic programming*. Princeton, NJ: Princeton University Press.
- Bermúdez-Otero, Ricardo. 2003. The acquisition of phonological opacity. In *Variation within Optimality Theory: Proceedings of the Stockholm Workshop*, ed. by Jennifer Spenader, Anders Eriksson, and Östen Dahl, 25–36. Stockholm: Stockholm University, Department of Linguistics.
- Berwick, Robert C., and Partha Niyogi. 1996. Learning from triggers. *Linguistic Inquiry* 27:605–622.
- Biró, Tamás. 2013. Towards a robust interpretive parsing: Learning from overt forms in Optimality Theory. *Journal of Logic, Language and Information* 22:139–172.
- Boersma, Paul. 1998. Functional phonology: Formalizing the interactions between articulatory and perceptual drives. Doctoral dissertation, University of Amsterdam.

- Boersma, Paul. 2001. Phonology-semantics interaction in OT, and its acquisition. In *Papers in experimental and theoretical linguistics*, vol. 6, ed. by Robert Kirchner, Wolf Wikeley, and Joe Pater, 24–35. Edmonton: University of Alberta.
- Boersma, Paul. 2003. Review of Tesar & Smolensky (2000): *Learnability in Optimality Theory*. *Phonology* 20:436–446.
- Boersma, Paul. 2007. Some listener-oriented accounts of h-aspiré in French. *Lingua* 117:1989–2054.
- Boersma, Paul. 2008. Emergent ranking of faithfulness explains markedness and licensing by cue. Rutgers Optimality Archive 954. Available at <http://roa.rutgers.edu/>.
- Boersma, Paul. 2009. Some correct error-driven versions of the Constraint Demotion Algorithm. *Linguistic Inquiry* 40:667–686.
- Boersma, Paul, and Bruce Hayes. 2001. Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry* 32:45–86.
- Boersma, Paul, and Joe Pater. 2016. Convergence properties of a gradual learning algorithm for Harmonic Grammar. In *Harmonic Grammar and Harmonic Serialism*, ed. by John J. McCarthy and Joe Pater, 389–434. Sheffield, UK: Equinox.
- Chomsky, Noam. 1957. *Syntactic structures*. The Hague: Mouton.
- Dell, François. 1970. Les règles phonologiques tardives et la morphologie dérivationnelle du français. Doctoral dissertation, MIT, Cambridge, MA.
- Dell, François. 1973. *Les règles et les sons*. Paris: Hermann.
- Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.
- Durand, Jacques, and Chantal Lyche. 2008. French liaison in the light of corpus data. *Journal of French Language Studies* 18:33–66.
- Eisner, Jason. 1997. Efficient generation in primitive Optimality Theory. In *Proceedings of the 35th Annual ACL and 8th European ACL*, 313–320. Available at <http://cs.jhu.edu/~jason/papers/eisner.acl97.pdf>.
- Ellison, T. Mark. 1994. Phonological derivation in Optimality Theory. In *Proceedings of the 15th International Conference on Computational Linguistics*, 1007–1013. Available at <http://www.aclweb.org/anthology/C94-2163>.
- Encrevé, Pierre. 1988. *La liaison avec et sans enchaînement: Phonologie tridimensionnelle et usages du français*. Paris: Seuil.
- Encrevé-Lambert, Marie-Hélène. 1971. A propos de l'élosion en français. *Rééducation orthophonique* 60: 245–251.
- Eychenne, Julien. 2006. Aspects de la phonologie du schwa dans le français contemporain: Optimalité, visibilité prosodique, gradience. Doctoral dissertation, Université de Toulouse–Le Mirail.
- Ford, Lester R. 1956. *Network flow theory*. Paper P-923. Santa Monica, CA: RAND Corporation.
- Gibson, Edward, and Kenneth Wexler. 1994. Triggers. *Linguistic Inquiry* 25:407–454.
- Gnanadesikan, Amalia. 1997. Phonology with ternary scales. Doctoral dissertation, University of Massachusetts, Amherst.
- Goldwater, Sharon, and Mark Johnson. 2003. Learning OT constraint rankings using a maximum entropy model. In *Variation within Optimality Theory: Proceedings of the Stockholm Workshop*, ed. by Jennifer Spenser, Anders Eriksson, and Östen Dahl, 111–120. Stockholm: Stockholm University, Department of Linguistics.
- Hamann, Silke, Diana Apoussidou, and Paul Boersma. 2011. Modelling the formation of phonotactic restrictions across the mental lexicon. In *CLS 45-1: The Main Session. Papers from the 45th Annual Meeting of the Chicago Linguistic Society*, ed. by M. Ryan Bochnak, Peter Klecha, Alice Lemieux, Nassira Nicola, Jasmin Urban, and Christina Weaver, 193–205. Chicago: University of Chicago, Chicago Linguistic Society.
- Hengeveld, Kees, and J. Lachlan Mackenzie. 2008. *Functional Discourse Grammar: A typologically-based theory of language structure*. Oxford: Oxford University Press.

- Jackendoff, Ray. 1997. *The architecture of the language faculty*. Cambridge, MA: MIT Press.
- Jarosz, Gaja. 2013a. Learning with hidden structure in Optimality and Harmonic Grammar: Beyond robust interpretive parsing. *Phonology* 30:27–71.
- Jarosz, Gaja. 2013b. Naive parameter learning for Optimality Theory: The hidden structure problem. In *NELS 40*, ed. by Seda Kan, Claire Moore-Cantwell, and Robert Staubs, 1–14. Amherst: University of Massachusetts, Graduate Linguistic Student Association.
- Keller, Frank. 2000. Gradience in grammar: Experimental and computational aspects of degrees of grammaticality. Doctoral dissertation, University of Edinburgh.
- Kiparsky, Paul. 1982. From cyclic phonology to lexical phonology. In *The structure of phonological representations, vol. 1*, ed. by Harry van der Hulst and Norval Smith, 131–175. Dordrecht: Foris.
- Kirchner, Robert. 1995. Going the distance: Synchronic chain shifts in Optimality Theory. Rutgers Optimality Archive 66. Available at <http://roa.rutgers.edu/>.
- Kirkpatrick, Scott, C. Daniel Gelatt, and Mario P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220:671–680.
- Legendre, Géraldine, Yoshiro Miyata, and Paul Smolensky. 1990. Can connectionism contribute to syntax? Harmonic Grammar, with an application. In *Papers from the 26th Regional Meeting of the Chicago Linguistic Society*, ed. by Michael Ziolkowski, Manuela Noske, and Karen Deaton, 237–252. Chicago: University of Chicago, Chicago Linguistic Society.
- Levelt, Clara. 1995. Unfaithful kids: Place of articulation patterns in early vocabularies. Talk presented at the University of Maryland, College Park.
- Magri, Giorgio. 2012. Convergence of error-driven learning algorithms. *Phonology* 29:213–269.
- Mohanan, K. P. 1981. Lexical phonology. Doctoral dissertation, MIT, Cambridge, MA.
- Moreton, Elliott, and Paul Smolensky. 2002. Typological consequences of local constraint conjunction. In *WCCFL 21: Proceedings of the 21st West Coast Conference on Formal Linguistics*, ed. by Line Mikkelsen and Chris Potts, 306–319. Somerville, MA: Cascadilla Press.
- Orgun, C. O. 1995. Correspondence and identity constraints in two-level Optimality Theory. In *WCCFL 14: Proceedings of the 14th West Coast Conference on Formal Linguistics*, ed. by José Camacho, Lina Choueiri, and Maki Watanabe, 399–413. Stanford, CA: CSLI Publications.
- Pater, Joe. 2009. Weighted constraints in generative linguistics. *Cognitive Science* 33:999–1035.
- Pater, Joe. 2016. Universal Grammar with weighted constraints. In *Harmonic Grammar and Harmonic Serialism*, ed. by John J. McCarthy and Joe Pater, 1–46. Sheffield, UK: Equinox.
- Prince, Alan, and Paul Smolensky. 1991. Optimality. Handout from the Arizona Phonology Conference.
- Prince, Alan, and Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Technical Report TR-2, Rutgers University Center for Cognitive Science, New Brunswick, NJ. Published, Malden, MA: Blackwell (2004).
- Riggle, Jason. 2004. Generation, recognition, and learning in Finite State Optimality Theory. Doctoral dissertation, UCLA, Los Angeles, CA.
- Riggle, Jason. 2009. Violation semirings in Optimality Theory. *Research on Language and Computation* 7:1–12.
- Schane, Sanford. 1968. *French phonology and morphology*. Cambridge, MA: MIT Press.
- Selkirk, Elisabeth. 1972. The phrase phonology of English and French. Doctoral dissertation, MIT, Cambridge, MA. Published, New York: Garland (1980).
- Smolensky, Paul. 1995. On the internal structure of the constraint component Con of UG. Handout from a talk given at the University of Arizona.
- Tesar, Bruce, and Paul Smolensky. 1993. The learnability of Optimality Theory: An algorithm and some basic complexity results. Ms., Department of Computer Science & Institute of Cognitive Science, University of Colorado at Boulder. Rutgers Optimality Archive ROA 2. Available at <http://roa.rutgers.edu/>.

- Tesar, Bruce, and Paul Smolensky. 1998. Learnability in Optimality Theory. *Linguistic Inquiry* 29:229–268.
- Tesar, Bruce, and Paul Smolensky. 2000. *Learnability in Optimality Theory*. Cambridge, MA: MIT Press.
- Tranel, Bernard. 1996. French liaison and elision revisited: A unified account within Optimality Theory. In *Aspects of Romance linguistics*, ed. by Claudia Parodi, Carlos Quicoli, Mario Saltarelli, and Maria Luisa Zubizarreta, 433–455. Washington, DC: Georgetown University Press.
- Viterbi, Andrew J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13:260–269.

Amsterdam Center for Language and Communication

Spuistraat 134

1012VB Amsterdam

The Netherlands

paul.boersma@uva.nl

j.w.vanleussen@uva.nl