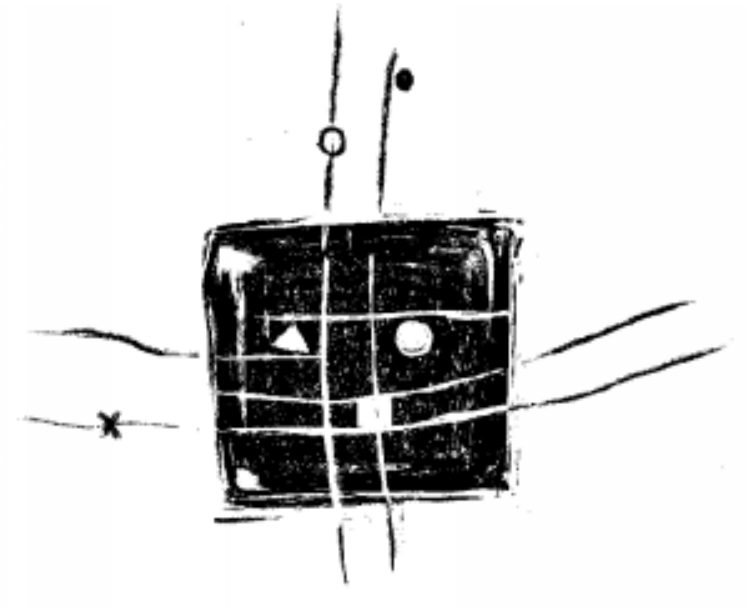
MARBURGER ARBEITEN ZUR LINGUISTIK



Nr. 6

Correspondence Theory: More Candidates Than Atoms in the Universe

Markus Walther

March 2001

Institut für Germanistische Sprachwissenschaft
Philipps-Universität Marburg
Wilhelm-Röpke-Str. 6A
D-35032 Marburg

e-mail: Markus.Walther@mailer.uni-marburg.de

---

archive: http://www.uni-marburg.de/linguistik/mal

Abstract

In Correspondence Theory (McCarthy & Prince 1995), elements of two strings (S_1, S_2) belonging to designated representational levels such as (input, output) may become associated through an arbitrary correspondence relation, while constraints like MAX, DEP evaluate such string pairs relative to the set of elementwise correspondences they find. Using some elementary calculations, I show that the number of candidates produced by correspondence-theoretic GEN quickly grows out of proportion, exceeding the estimated amount of atoms in the entire universe for strings S_1 of length greater 6 under conservative upper bounds for amount of epenthesis, reduplication and allophonic inventory size. This finding suggests that it is hopeless to do realistic manual verification of correspondence-theoretic analyses, while leaving open suitable restrictions of the formal power of correspondence, as e.g. proposed by Primitive Optimality Theory (Eisner 1997). While phonology may not count, phonologists clearly should sometimes.

Contents

1 Introduction

Correspondence Theory (McCarthy & Prince 1995; Kager 1999) (henceforth: CT) is an increasingly popular extension of classical Optimality Theory (Prince & Smolensky 1993) in which the notion of *correspondence* between elements of two strings plays a central role.[1] Historically, this notion first came up in reduplicative morphology, where theorists wanted to talk about the relation between corresponding segments of the base and the reduplicant (McCarthy & Prince 1994), but the model has been generalized almost immediately (McCarthy & Prince 1995), in particular to input-output correspondence. Input-output correspondence may be safely assumed to be the standard instantiation of CT in the sense that constraints will need to reference it in the most basic analytic scenarios, i.e. those that require neither formal means for handling reduplication nor means for describing output-output relations. Hence, we will focus the following discussion on this type of correspondence, though nothing crucially hinges on this decision.

CT abandons the monostratal conception of its predecessor in favour of a two-level model. In the old conception – the so-called 'containment' variety of OT – the underlying representation (UR) was merely enriched with prosodic structure, underparsing marks and epenthetic segments, but the underlying input was guaranteed to be contained in every candidate. In CT, however, this guarantee is no longer upheld. In fact, the surface representation (SR) produced by GEN may now be any (prosodically decorated) string drawn from the allophonic inventory of the world's languages, even if there is no discernible similarity to the underlying form, and it is the sole responsibility of the constraints to ensure that the correct output is produced.

In order to relate UR and SR, CT introduces the concept of a *correspondence relation* $\mathcal{R}$, whose purpose in the input-output specialization of correspondence is to record pairings of individual elements from UR and SR. Here is the original definition:

> Given two strings S_1 and S_2, **correspondence** is a relation $\mathcal{R}$ from the elements of S_1 to those of S_2. Elements $\alpha \in S_1$ and $\beta \in S_2$ are referred to as **correspondents** of another when $\alpha \mathcal{R} \beta$. (McCarthy & Prince 1995, 14, *emphasis in original*)

[1]This work has been funded by the German research agency DFG under grant WI 853/4-2. Many thanks to the participants of the Marburg phonology discussion group for helpful comments and criticism, in particular Birgit Alber, Stefan Girgsdies, Wolfgang Kehrein, and Richard Wiese. All remaining errors and shortcomings are mine.

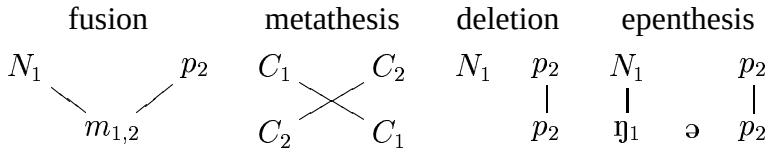
According to the inventors of CT, elements are just (tokens of) segments, although they add that generalization to "moras, syllables, feet, heads of feet, as well as tones and even distinctive features or feature nodes" (McCarthy & Prince 1995, 14) should be straightforward. Constraints are now empowered to evaluate entire two-level mappings between subparts of UR and SR instead of the former SR-only mode of strict output orientation, and they may distinguish between correspondent and non-correspondent segment pairs when evaluating such mappings. The family of correspondence relations $\mathcal{R}_{UR_i,SR_j}$ induced by the set of (UR_i,SR_j) pairs is best thought of as a hidden part of the universal GEN function, which continues to be parametrized by input UR (= S_1) only:

> Each candidate pair (S_1,S_2) comes from Gen equipped with a correspondence relation between S_1 and S_2 ... There is also a correspondence relation for each (I,O) candidate-pair. Indeed, one can simply think of Gen as supplying correspondence relations between S_1 and *all possible structures over some alphabet.* (McCarthy & Prince 1995, 14f, *emphasis added*)

If instead $\mathcal{R}$ existed outside of GEN, i.e. if something like a set of tailor-made correspondence relations $\mathcal{R}_{i,j}$ were stored within each lexical entry $(UR_i, \{(\mathcal{R}_{i,1}, SR_1), (\mathcal{R}_{i,2}, SR_2), \ldots, (\mathcal{R}_{i,k}, SR_k)\}), 0 < k < \infty$, one would have to face a massive increase in idiosyncratic information, which seems hard to defend.

To illustrate the freedom introduced by correspondence, here is a graphical rendering of some linguistically useful correspondences from a real analysis (Kager 1999, 62-69):

(1)



Note how this selection encompasses many-to-one correspondences (fusion), correspondences invalidating precedence relationships across tiers (fusion,metathesis), absence of correspondence with the output (deletion) and input (epenthesis), in addition to straightforward precedence-preserving one-to-one correspondences (parts of deletion, epenthesis). Also displayed are instances

of corresponding segments that miss perfect identity (both correspondences in fusion, leftmost correspondence in epenthesis).

2 How many correspondences and candidates?

In this section, I will develop some simple formulae to calculate the number of correspondences and resulting candidates. To make it easier to understand, the presentation will be broken down into a number of intermediate steps.

Just two strings: how many correspondences? We are given two finite-length strings S_1, S_2, and ask ourselves: how many ways are there to draw correspondence lines between elements of S_1 and S_2? Recall that CT places no restrictions whatsoever on multiple corresponce and also allows for the possibility that any individual element may not correspondent to any counterpart. The reasoning evidently is that the constraints should see the richest possible set of candidates – GEN should not be restrictive and take over some work of CON.

Simple combinatorics then lead us to the conclusion that any element of string S_1 can combine with every subset of elements in the string S_2. Note that we can freely convert between string and set representations by representing each string segment x as a tuple $(x, position\_of\_x)$ in the set representation, cf. McCarthy & Prince (1994, p.7,fn.8). For example, with string positions starting at 1 the string aba is equivalently represented by a set $s = \{(b, 2), (a, 3), (a, 1)\}$. Designating the lengths of $|S_1| = n$ and $|S_2| = m$ and using the fact that for a given set s with cardinality m the set of all subsets of s has cardinality 2^m, we get the following formula:

$$(2) \quad numberOfCorrespondences(n, m) = \underbrace{2^m \cdot 2^m \cdot \ldots \cdot 2^m}_{n\ times} = 2^{m \cdot n}$$

How many output strings for given input? To answer the question we need to look at two factors. The first factor encompasses possible ways to increase the length of an input string S_1. There are two obvious subfactors, epenthesis and reduplication. Hence, let us posit corresponding variables $amountOfEpenthesis, amountOfRed$. The first variable is a natural number that designates the *maximum* number of epenthetic segments[2] per candidate string

[2]The term is slightly imprecise, as it is immaterial whether those segments are actually non-correspondent (true epenthesis) or not (e.g. gemination, vowel lengthening). All that matters is that they cause the length of S_2 to increase.

S_2. A reasonable default value for most practical cases would seem to be 2. The second variable is a natural number which is 1 for no reduplication, 2 for a maximum of one total reduplication of S_1, 3 for a total triplication of S_1 and so forth. Again, a reasonable default value would be 2. Putting things together, we get:

(3) $maxOutputLength(n) = n \cdot amountOfRed + amountOfEpenthesis$

The preceding figure gives the maximum length of S_2 as a function of the length n of S_1. Note that we must not think of the length increase as being counterbalanced by the peculiar nature of the factors contributing to this increase: due to universality, epenthetic and reduplicated segments of *any* segmental quality will be posited by GEN, with only the constraints being responsible for restricting reduplicative or epenthetic identity in appropriate ways. Also, one must not be mislead by the fact that the set of correspondence relations for the 'Full Model' of reduplication depicted in (McCarthy & Prince 1995, ex. (145)) does not explicitly draw an arrow between input and output. Not only is I-O correspondence mentioned in the paragraph preceding their example (145), but also the output level surely must continue to function as the sole level over which phonetic interpretation is defined – reduplicated word forms are pronounced after all. Thus, restricting the present discussion to I-O correspondence *and* at the same time including a reduplicative factor contributing to an overall length increase does make sense.

However, we are not finished yet. At each position in S_2 we have a choice between all *crosslinguistically possible allophones*. Why is this? Simply because GEN is again understood as being universal, hence should not prejudice against any possible segmental correspondent in the surface form, leaving that work entirely to the constraints. Obviously it is not a simple task to determine the maximum number of crosslinguistically possible allophones. However, a conservative lower bound[3] may be derived from the number of distinct segments mentioned in the UPSID database of over 400 languages (Maddieson 1984), which yields a count of 913. Thus, maa, a constant denoting the maximum amount of allophones, will receive the value of 913 as the default.

To complete the picture, we only need to observe that, due to the existance of truncation processes in the world's languages, all strings of length *shorter* than $maxOutputLength(n)$ also need to be generated. For a fixed length k we get maa^k different strings, because we have a choice between one of maa allophones

[3]The lower bound is a conservative estimate because (i) UPSID only samples inventories from $\sim$ 450 languages, but there are at least 6000 of them and (ii), languages may well exhibit allophones with phonologically-controlled distribution which are not phonemic in any language.

at each of the k independent positions. Summing over all lengths, we therefore arrive at the following formula for the number of output strings:

$$numOutputs = maa^0 + maa^1 + \cdots + maa^{maxOutputLength(n)} \tag{4}$$

One input, many outputs and correspondences: what's the total? It is time to put the pieces together, amalgamating formulae (2) and (4). We learned from (2) how many correspondences there are for two fixed strings S_1, S_2. Equation (4) showed us how many S_2 will result from a single S_1. Because the correspondence possibilities are independent of the concrete segmental identities in each of the many S_{2_i} strings, but obviously dependent on their length – segments cannot correspond to nonexisting material –, we now need to multiply both quantities *per length class*. This gives us the final formula for the total number of corresponding string-string pairs produced by GEN (summands written according to increasing length class):

$$\begin{aligned} numCandidates(n) = {} & 2^{n\cdot 0} \cdot maa^0 + 2^{n\cdot 1} \cdot maa^1 + 2^{n\cdot 2} \cdot maa^2 + \ldots \\ & + 2^{n\cdot maxOutputLength(n)} \cdot maa^{maxOutputLength(n)} \end{aligned} \tag{5}$$

Note that all we have done so far is to calculate the number of undecorated string-string pairs under correspondence. This is only a lower bound on the actual number of candidates because in real candidates an output string is only the terminal yield of a prosodic tree, which adds subsyllabic roles and/or moras, syllable structure, foot structure, prosodic words, and possibly higher categories. Crucially, the assignment of prosodic trees to output strings is again many-to-one, as empty categories, different syllabifications of the same string, subtrees violating the strict layer hypothesis of the prosodic hierarchy etc. have all been defended and used in actual analyses. As a consequence, these alternative assignments need to be produced by universal GEN and we would again have to multiply the quantity in (5) by some unknown but probably large amount to arrive at the real number of candidates.

To give the reader both a feeling for the multitude of possibilities that arises under correspondence-theoretic GEN and help her verify the formula in (5) for a concrete example, figure 1 on page 16 shows a sample tabulation for an unrealistically small allophone alphabet Σ = {a,b} and an input string of length 2 with no increase due to epenthesis or reduplication. The tabulation is exhaustive, each candidate is unique, and the resulting number of 73 candidates is just what the formula predicts.

Any realistic numbers? To compute values according to (5), we need to fix values for the free parameters $amountOfEpenthesis, amountOfRed, maa$. A reasonable first choice is to take the default values from above: $amountOfEpenthesis = 2, amountOfRed = 2, maa = 913$. Here then are some values for $numCandidates(n)$ with the just-mentioned parameter values, tabulated for input strings S_1 of increasing length n:

(6) `tabulate_num_candidates(7)`

```
n = 1 =>  11123488168255
        order =  10^14
n = 2 =>  2373031750999377744277
        order =  10^22
n = 3 =>  810112960232046271248047918868l
        order =  10^31
n = 4 =>  44252451301762855290066685306456093887l313
        order =  10^42
n = 5 =>  3867799344454428986946948265606947514339496\
          90953561633
        order =  10^54
n = 6 =>  5409012110360452391275004950726834474931984\
          207274730971903465813057
        order =  10^67
n = 7 =>  1210307351317901990540944081414944838073936\
          025991359921232840158498455912075364481
        order =  10^82
```

These values were computed using a simple script fed into the arbitrary-precision calculator *bc*, which is available on many UNIX-like systems. The script is documented in appendix A.

To put those figures into perspective: it is assumed that the number of atoms forming the observable mass of the entire universe is in the order of 10^{79} (see e.g. http://www.sunspot.noao.edu/PR/answerbook/universe.html#q70). Thus, with only length-7 inputs we have already surpassed that order!

Because of the thoroughly exponential nature of formula (5), nothing really hinges on the particular choice of default parameters. Even the most restricted case possible, which has an absurd output inventory consisting of a single allophone and a total ban on additional epenthesis or reduplication, displays the

exponential blowup, ending up well above 10^{79} already for length-17 inputs like English *industrialization* [ɪndʌstrɪəlɑizeɪʃn]. The reason for this behaviour is evident from inspection of (5): the formula contains two kinds of exponential terms, and shrinking the allophone inventory size to 1 has eliminated only one of them. The exponential factors modelling the number of correspondence patterns remain. In other words, the existence of free correspondence in itself is sufficient to cause the dreaded exponential blowup.

3 What are the consequences?

Checking of correspondence-theoretic analyses by hand is impossible. A naive enumeration of candidates under CT is simply infeasible, given the sheer number of candidates that arise even for short inputs. But that is the current standard practice – show some tableaux with number of candidates in the order of 10^1 at most, and conclude that the claim to winner status of some candidate marked with the hand symbol is valid.

One might respond by trying to improve on naive enumeration – let us devise some clever enumeration scheme. The question then becomes: Can we devise a universal sorting order for candidate, so that lazy production of candidates – only change some correspondence start pattern if demanded by the constraints, with the change going from minimal to maximal deviance along some canonical route – is a practical possibility? It seems not. Suppose someone devises a constraint which favours a weird selection of correspondence pattern that may be somewhat hard to grasp for humans, but demonstratably solves an empirical puzzle. If this constraint is top-ranked, the candidates that do not violate it will have to come first in the constraint-induced sorting order, even though they may be widely separated in the easy-to-grasp canonical sorting order. In other words, the analyst again has to use her intuition about which candidates to include in the winner computation instead of carrying out a mechanical proof based on an easily memorized canonical sorting order, because the latter is not guaranteed to lead to an optimality proof in a reasonably short time frame for all possible constraints. This dependence of sorting on the action of constraints therefore destroys the hope of coming up with a once-and-for-all sorting order that is valid across analyses and rerankings.

Does SPE-type derivationalism face the same problems? No. Each SPE rule produces some local modifications of a single phonological object string, which are typically very easy to verify. Abbreviatory conventions and special rule ap-

plicability conditions do complicate the picture somewhat, but without invalidating the general point: it is comparatively easy to check that all the rule-induced mappings in a derivational cascade from UR to SR are indeed licensed by the respective rules.

Here is an example illustrating easy verifiability of SPE rules. Consider the two rules under (7), taken from Kaplan & Kay (1994):

(7)

1. N $\rightarrow$ m / ___ [+labial]
2. N $\rightarrow$ n

Let us take the strings *iNpractical* and *iNtractable* as inputs. Then, applying the rules in the order indicated reduces to first scanning for a length-2 substring *N[+labial]*, which – with appropriate definition of the feature bundle involved as, say, $[+labial] \equiv \{b, p, f, v\}$ – yields a match at string position 2 for *iNpractical* and no match for *iNtractable*. Having carried out the substitution $N \rightarrow m$ at position 2 of the first string, we rescan both strings for the length-1 substring *N*, which now yields a match followed by substitution $N \rightarrow n$ only for *iNtractable* at position 2. The output strings thus produced are *impractical* and *intractable*. Clearly, rule application is simple enough to be exhaustively checked by hand for most cases.

But better yet, even supposing that rule complexity did look overwhelming in a particular instance, we still have the mechanical rule translation strategy that is rigorously defined in Kaplan & Kay (1994). Each rule individually translates into a finite-state transducer, a mathematically well-understood formal device. Next, entire rule cascades involving an *arbitrary* fixed number of rules are collapsed into a *single* transducer that encodes the overall input-to-surface mapping by simply connecting individual transducers with the mathematical *composition* operator.[4] Since each of these steps – including the final application to a given input or set of inputs – is also practically computable, one can simply delegate the analysis-checking task to a computer program that implements the rule-translation-and-application procedure (e.g. by using the *FSA Utilities* software of van Noord 1997). Assuming that the software faithfully reproduces the mathematical descriptions, computerized rule checking is nothing less than an exhaustive mathematical proof of the claims a given rule-based analysis makes with respect to the correct output strings. Thus, *pace* McCarthy (1999, 392), it is not the case that, especially

[4]Kaplan & Kay give full details of the finite-state transducers corresponding to our rules in (7) and also show the composition transducer collapsing serial application of rule 2 after rule 1.

for more than two interacting rules, the "broad consequences of rule ordering and serialism remain largely unexamined": the problem was solved in a mathematically and computationally sound way many years ago.

Does containment-type classical OT have similar problems? It seems that classical OT faces a much less severe blowup in the number of candidates as a function of input length. There are at least two factors involved: underparsing and epenthesis. For a given string S_1, $|S_1| = n$, there are 2^n candidate output strings to due underparsing: each subset of S_1 may be underparsed (including the empty set, which means no underparsing at all). Epenthesis is interpreted to work on top of this: for a given abstract epenthetic element, each possibly underparsed candidate of length n can have it inserted at one of $n + 1$ positions. If there is a maximum of k epenthetic positions and the entire range of inserting 0,1, ... ,k epenthetic elements is considered, we get a multiplicative factor of $1 + (n+1)^1 + (n+1)^2 + \cdots + (n+1)^k$. The total is therefore defined as follows:

(8)
$$numCandidatesClassical = 2^n \cdot (1 + (n+1)^1 + (n+1)^2 + \ldots + (n+1)^{amountOfEpenthesis})$$

This number rises much more slowly (*amountOfEpenthesis*=2):

(9)

```
n = 1 =>  14
        order =  10^2
n = 2 =>  52
        order =  10^2
n = 3 =>  168
        order =  10^3
n = 4 =>  496
        order =  10^3
n = 5 =>  1376
        order =  10^4
n = 6 =>  3648
        order =  10^4
n = 7 =>  9344
        order =  10^4
n = 8 =>  23296
```

```
        order =  10^5
n = 9 =>  56832
        order =  10^5
n = 10 =>  136192
        order =  10^6
```

To reach 10^{79}, one needs a very long string of length n=244. Of course, containment theory as originally defined has several deficits, amongst them the abstractness of epenthesis, where a concrete segmental identity is not defined for epenthetic segments, and the unclear status of both reduplication and metathesis in that model. Also, underspecified input segments would result in an increase in the number of candidates, because of the various ways of fleshing them out to fully specified candidates. Finally, longer inputs would certainly favour a computational evaluation as well: e.g. (Karttunen 1998) noted that his simplified definition of GEN already produced 1,7 million candidates for input /abracadabra/. However, for checking of analyses *per se* – our main concern here – containment theory remains definitely preferrable to CT.

Are there other mathematical or computational strategies available? Perhaps, but this needs more research. One might try to resort to an *intensional formal description* of both candidate sets and CT constraints. "Don't literally enumerate at all, just describe the possibilities using some compact formal notation" would be the slogan here.

As soon as this path is followed, one of two possibilities arise: (A) conduct mathematical proofs based on the intensional formal descriptions, or (B) automate the proofs via a suitable computational evaluation procedure. Possibility (A) seems ill-suited to the needs of the practicing phonologist, even if proof schemata could be constructed that would somehow abbreviate the difficult manual construction of proofs by only requiring the filling-in of certain parts that would vary as a function of the constraints and rankings used. Possibility (B) immediately invokes the follow-up question of a suitable formal basis for formalizing the constraints and GEN. So far, concrete proposals in this domain have all used finite-state automata to represent CON and GEN, for good reasons. Finite-state automata are very attractive in terms of mathematical clarity and efficient processing. The drawback for present purposes is that such automata cannot systematically represent the token identity required in differentiating between corresponding segments that may have the same segmental quality: a_1 and a_2 may have different corre-

spondents, as indicated by the subscripts. Finite-state automata, however, cannot manage more than a finite, *a priori* fixed number of such differentiating indices (or equivalent devices), which is problematic because in general no principled upper bound on the length of inputs seems to be available. Of course, one is free to reject the proven finite-state basis for CON and GEN formalization, but then the burden of proof shifts to the rejector to come up with an adequate formal and computational treatment of CT using whatever more powerful formalism is proposed instead.

At present, a wiser strategy seems to be to stick to the finite-state framework and seek ways to reduce the excessive power of arbitrary correspondence. A good candidate to start with might be Primitive Optimality (Eisner 1997; Albro 1997) (henceforth: OTP). We are generally interested in OTP because it represents GEN and constraints with finite-state automata, thus possessing a sound formal and computational basis. More to the point, OTP favours a mode of representation where phonological information is arranged in an autosegmental-like fashion, being split among a fixed number of tiers, including technical tiers differentiating between underlying and surface material or inserted versus deleted substrings. However, compared to the autosegmental finite-state treatment of (Bird & Ellison 1994), there are no explicit association lines; rather, correspondence-as-association is construed directly via temporal overlap of temporally extended 'autosegments' or 'gestures'. The job of GEN, CON and EVAL is to flesh out the temporally underspecified input in a way that produces another finite-state automaton representing the winner(s). Leaving matters at that, the possibilities resulting from this implicit-correspondence-through-temporal-overlap model would be greatly restricted. This is because in the original CT proposal crossed correspondences are valid, but both Bird & Ellison's and Eisner's temporal semantics of representational diagrams turn any violation of the no-crossing constraint into an empty automaton, leaving essentially the two options of local spreading and one-to-one correspondence.

Unfortunately, this might be restricted just a little too much, because then metathesis and reduplication become impossible to express. A general treatment of crossing correspondences requires not only greater-than-regular formal power for GEN, e.g. Multiple Context Free Grammars (Albro 2000), but also non-finite-state constraints due to the token identity problem mentioned above. However, a representational trick might offer at least some help here within the confines of finite-state OTP (p.c. Dan Albro): for each underlying input string UR on its own tier, mechanically add a second, derived, tier UR_{perm} representing all possible permutations of UR. Since tiers are just automata, and the set of permuta-

tions is representable as an automaton, too, this much certainly seems possible within a finite-state framework. One would then have the freedom of making constraints sensitive to either the original UR tier or the derived UR_{perm} tier when evaluating correspondence-as-overlap against a given surface tier. Since the two UR tiers are systematically related, a constraint can now assess, say, a formally non-crossing correspondence between SR and segments in UR_{perm}, while the direct correspondences using UR would cross. While this allows for scenarios like /pat/$_{UR}$ – /tap/$_{UR_{perm}}$ – [ta?]$_{SR}$, it still does not give us the general ability to distinguish between segments that only differ in position, like /pa$_1$a$_2$/$_{UR}$ – /pa$_2$a$_3$/$_{UR_{perm}}$ – in other words, the token identity problem remains.

In sum, a promising route to modify CT's problematic notion of arbitrary correspondence might be to revert to the autosegmental-temporal semantics of representations and rule out both formally crossing correspondences and overt labelling of correspondence indices, as proposed by Primitive Optimality Theory. It remains to be seen whether the phonological community will adopt OTP as a new reference framework.

What about other types of correspondence? There is not much to say here, because other instantiations like base-reduplicant or input-reduplicant correspondence are clearly also correspondences in the formal sense. Thus, they can only add their combinatory potential to the set of possibilities arising through 'basic' I-O correspondence. The situation is in principle similar with output-output (O-O) correspondence, where surface representations of individual words or word forms directly correspond with one another. However, if I-O correspondence were altogether eliminated in a radical, massively parallel O-O-only model, the ensuing new calculations might indeed lead to new results. However, since the number of word forms is infinite in practice in languages like German and Turkish (free compounding, iterated use of affixes), and infinite in theory in every language due to productivity, it is very unclear at present how precisely a pure O-O model would handle those important aspects of language, let alone whether such a model is practically computable at all. Even if it could be shown to be formally welldefined under those circumstances, the ensuing numbers of correspondences would be infinite in the limit, thus only worsening the manual checking problem that this paper has highlighted.

4 Conclusion

I have shown that correspondence theory (McCarthy & Prince 1995) generates a horribly large amount of candidates which quickly surpasses the amount of atoms in the entire universe even under most restrictive assumptions. This finding renders the standard practice of eyeballing a few hand-selected candidates a highly questionable enterprise with doubtful scientific status. Whatever the problems of the older derivational theories as compared to the correspondence-theoretic framework may be, they are rendered insignificant in the face of the sheer impossibility of manual verification of concrete analyses in the latter. Further research involving modification of central assumptions of correspondence theory and/or computerized analysis checking seems inevitable to restore at least some of the credibility of this extension of OT. While phonology may not count, phonologists clearly should sometimes.

References

Albro, D. (1997). Evaluation, Implementation, and Extension of Primitive Optimality Theory. Master's thesis, Department of Linguistics, UCLA.

Albro, D. (2000). Taking Primitive Optimality Theory Beyond the Finite State. In: J. Eisner, L. Karttunen & A. Thériault (Eds.), *SIGPHON 2000 'Finite-State Phonology' – Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology*, Luxembourg, 57–67. 6th August 2000, COLING 2000.

Bird, S. & T. M. Ellison (1994). One-Level Phonology. *Computational Linguistics 20*(1), 55–90.

Eisner, J. (1997). Efficient generation in primitive Optimality Theory. In: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Association for Computational Linguistics*, Madrid.

Kager, R. (1999). *Optimality Theory*. Cambridge University Press.

Kaplan, R. & M. Kay (1994). Regular models of phonological rule systems. *Computational Linguistics 20*(3), 331–78.

Karttunen, L. (1998). The Proper Treatment of Optimality in Computational Phonology. In: *Proceedings of FSMNLP'98. International Workshop on*

Finite-State Methods in Natural Language Processing, Bilkent University, Ankara, 1–12. (CLA-9804002,ROA-258-0498).

Maddieson, I. (1984). *Patterns of sound*. Cambridge University Press.

McCarthy, J. & A. Prince (1994). The Emergence of the Unmarked: Optimality in Prosodic Morphology. In: M. Gonzalez (Ed.), *Proceedings of the North-East Linguistics Society*, Vol. 24, Amherst, MA, 333–379. Graduate Linguistic Student Association. (ROA-13).

McCarthy, J. & A. Prince (1995). Faithfulness and reduplicative identity. In: J. Beckman, L. W. Dickey & S. Urbanczyk (Eds.), *Papers in Optimality Theory*, Vol. 18 von *University of Massachusetts Occasional Papers in Linguistics*, 249–384. Amherst, MA: Graduate Linguistic Student Association. (ROA-60).

McCarthy, J. J. (1999). Sympathy and phonological opacity. *Phonology 16*(3), 331–400.

Prince, A. & P. Smolensky (1993). Optimality Theory. Constraint Interaction in Generative Grammar. Technical report RuCCS TR-2, Rutgers University Center for Cognitive Science.

van Noord, G. (1997). FSA Utilities: A toolbox to manipulate finite-state automata. In: D. Raymond, D. Wood & S. Yu (Eds.), *Automata Implementation*, Vol. 1260 von *Lecture Notes in Computer Science*, 87–108. Springer Verlag. (Software under `http://grid.let.rug.nl/~vannoord/Fsa/`).

A Script for numCandidates calculation using *bc*

```
# (c) 2001 by Markus Walther, University of Marburg

define num_candidates_recursive (len, n, maa) {
        if (len < 1) return (1);
        return (num_candidates_recursive(len-1, n, maa) \
                + (2^(n*len))*(maa^len));
        }

define num_candidates(n,redupfactor,epenthesisfactor,maa) {
        len = n*redupfactor+epenthesisfactor;
        return ( num_candidates_recursive(len,n,maa) );
}

define tabulate_num_candidates (max) {
        for ( i=1; i<=max; i++ ) {
        result = num_candidates(i,2,2,913);
        print "n = ", i, " =>  ", result;
        print "\n";
        print "        order =  10^", length(result);
        print "\n";
        }
        print "\n";
}
```

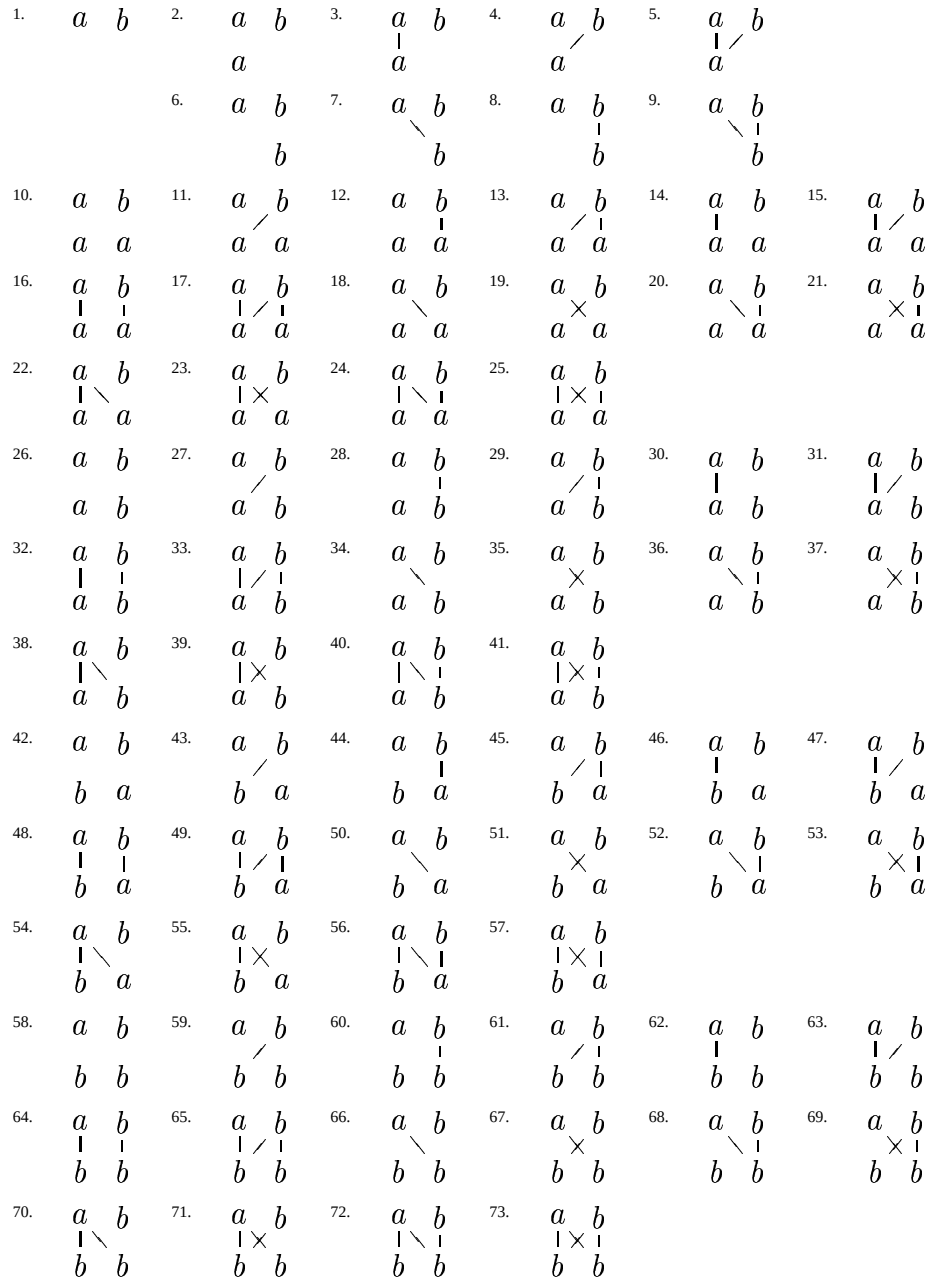


Figure 1: GEN output with correspondences for S_1 = ab, Σ = {a,b}, $amountOfEpenthesis = 0, amountOfRed = 1, maa = 2$