April 26, 2006

**CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION**

April 26, 2006

1

The Insufficiency of Paper-and-Pencil Linguistics: the Case of Finnish Prosody

Lauri Karttunen

1.1 Introduction

It is a basic scientific practice to examine a limited amount of data in the light of some theoretical framework and to develop an analysis that accounts for the primary facts and extends to unseen data. If the predictions are correct, the analysis stands and lends further support for the framework in which it is conceived.

This paper focuses on a case where the analysis turns out to be wrong. It highlights the need for the formalization and computational implementation of linguistic theories. Paper-and-pencil methods are insufficient to test a theory with real data. The problem is particulary acute for optimality theory (Prince and Smolensky, 1993, Kager, 1999, McCarthy, 2002). The ot framework assumes two levels of representation, a set of *inputs* and, for each input, a possibly infinite set of *outputs*. The mapping between the two levels is subject to a set of ranked constraints. It is typically the case that no output candidate satisfies all the constraints. A winning candidate is the one that incurs the fewest violations of the most highly ranked constraint still in play that cannot be satisfied by any of the other surviving output candidates.

We consider two closely related ot analyses of Finnish prosody by Elenbaas (1999) and Kiparsky (2003). In both cases the input consists of a sequence of phonemes and the outputs are sequences of metrical

feet that consist of syllables with stress marks, as shown in (1) for the input *opiskelijakin*.

(1) (ó.pis).(kè.li).(jà.kin) 'even the student' (Sg. Nom.)

Here the acute accent indicates primary stress and the two grave accents mark secondary stress. Periods mark syllable boundaries and feet are enclosed in parentheses.

In general, Finnish prosody is trochaic with the main stress on the first syllable and a secondary stress on every other following syllable. Finnish also has a ternary stress pattern that surfaces in words where the stress would fall on a light syllable that is followed by a heavy syllable. A light syllable ends with a short vowel (*ta*); a heavy syllable ends with a coda consonant (*jat*, *an*) or a long vowel (*kuu*, *aa*) or a diphthong (*voi*, *ei*). Example (2a) shows the correct ternary prosody for the input *rakastajattarenako* 'mistress' (Sg. Ela., QP).

(2) a. (rá.kas).ta.(jàt.ta).(rè.na).ko
b. *(rá.kas).(tà.jat).(tà.re).(nà.ko)

(2b) shows the binary stress pattern that is incorrect because of the (tà.jat) foot where the stress falls on a light syllable followed by a heavy syllable. The initial (rá.kas) syllable in (2a) is actually a violation of the same stress constraint but it is allowed by a more highly ranked constraint, specific to Finnish, that requires the main stress on the initial syllable.[1]

It has been claimed that the ternary prosodic pattern arises naturally, in the context of Optimality Theory (OT), from the interaction of independently motivated optimality constraints such as *Lapse and StressToWeight. The idea has its origins in Hanson and Kiparsky (1996). It has been explored in depth in the Ph.D. thesis of Nine Elenbaas (1999) and summarized in the articles by Elenbaas and Kager (1999) and Kiparsky (2003).

This paper formalizes the Elenbaas and Kiparsky analyses in finite-state terms using lenient composition (Karttunen, 1998) to prune the candidate set. It shows that the two OT analyses yield incorrect results in cases such as (3).

(3) *(ká.las).te.(lè.mi).nen 'fishing' (Sg. Nom.)

The specific conclusion is that the explanation for the ternary meter offered by Elenbaas and Kiparsky fails systematically for certain input patterns, but a more general point is that OT phonology badly needs

[1] Kiparsky and Elenbaas treat the third syllable of a dactyl as extrametrical, that is, (rá.kas).ta. instead of (rá.kas.ta). This decision of not recognizing a ternary foot as a primitive is of no consequence as far as the topic of this paper is concerned.

computational support. It is difficult to get globally correct results from a handful of examples with the traditional tableau method.

1.2 OT Constraints for Finnish Prosody

Under Kiparsky's analysis (p. 111), the prosody of Finnish is characterized by the system in (4). The constraints are listed in the order of their priority.

(4) a. *Clash: No stresses on adjacent syllables.
b. Left-handedness: The stressed syllable is initial in the foot.
c. Main Stress: The primary stress in Finnish is on the first syllable.
d. FootBin: Feet are minimally bimoraic and maximally disyllabic.
e. *Lapse: Every unstressed syllable must be adjacent to a stressed syllable or to the word edge.
f. Non-Final: The final syllable is not stressed.
g. Stress-To-Weight: Stressed syllables are heavy.
h. License-σ: Syllables are parsed into feet.
i. All-Ft-Left: The left edge of every foot coincides with the left edge of some prosodic word.

Elenbaas (1999) and Elenbaas and Kager (1999) give essentially the same analysis except that they replace Kiparsky's Stress-To-Weight constraint with the more specific one in (5).

(5) *(L̀H): If the second syllable of a foot is heavy, the stressed syllable should not be light.

Kiparsky, Elenbaas and Kager construct the ranking of these constraints by considering all possible output candidates for a fair number of multisyllabic words. They show that only the ordering in (4) yields the right outcome. For example, the fact that (2a) is preferred over (2b) indicates that License-σ is dominated by Stress-To-Weight (or *(L̀H)). The contrast between (6a) and (6b) indicates that Stress-To-Weight in turn is dominated by *Lapse.

(6) a. (rá.vin).(tò.lat) 'restaurant' (Pl. Nom.)
b. *(rá.vin).to.lat

1.3 Finite-State Approximation of OT

As we will see shortly, classical OT constraints such as those in (4) and (5) are REGULAR (= RATIONAL) in power. They can be implemented by finite-state networks. Nevertheless, it has been known for a long

time (Frank and Satta, 1998, Karttunen, 1998, Eisner, 2000) that OT as a whole is not a finite-state system. Although the official OT rhetoric suggests otherwise, OT is fundamentally more complex than finite-state models of phonology such as classical Chomsky-Halle phonology (Kaplan and Kay, 1994) and Koskenniemi's two-level model (Koskenniemi, 1983). The reason is that OT takes into account not just the ranking of the constraints but the number of constraint violations. For example, (7a) and (7b) win over (7c) because (7c) contains two violations of *Lapse whereas (7a) and (7b) have no violations.[2]

(7) a. (ér.go).(nò.mi).a 'ergonomics' (Nom. Sg.)
b. (ér.go).no.(mì.a)
c. (ér.go).no.mi.a

Furthermore, for GRADIENT constraints such as All-Ft-Left, it is not just the number of instances of non-compliance that counts but the SEVERITY of the offense. Candidates (7a) and (7b) both contain one foot that is not at the left edge of the word. But they are not equally optimal. In (7a) the foot not conforming to All-Ft-Left, (nò.mi), is two syllables away from the left edge whereas in (7b) the noncompliant (mì.a) is three syllables away from the beginning. Consequently, (7b) with three violations of All-Ft-Left loses to (7a) that only has two violations of that constraint.

If the number of constraint violations is bounded, the classical OT theory of Prince and Smolensky (1993) can be approximated by a finite-state cascade where the input is first composed with a transducer, GEN, that maps the input to a set of output candidates (possibly infinite) and the resulting input/output transducer is then "leniently" composed with constraint automata starting with the most highly ranked constraint. We will use this technique, first described in Karttunen (1998), to implement the two OT descriptions of Finnish prosody. The key operation, LENIENT COMPOSITION, is a combination of ordinary composition and PRIORITY UNION (Kaplan and Newman, 1997).

The basic idea of lenient composition can be explained as follows. Assume that `R` is a relation, a mapping that assigns to each input form some number of outputs, and that `C` is a constraint that prohibits some of the output forms. The lenient composition of `R` and `C`, denoted as `R .O. C`, is the relation that eliminates all the output candidates of a given input that do not conform to `C`, provided that the input has at least one output that meets the constraint. If none of the output candidates of a given input meet the constraint, lenient composition

[2]It is important to keep in mind that the actual scores, 0 vs. 2, are not relevant. What matters is that (7a) and (7b) have **fewer** violations than (7c).

spares all of them. Consequently, every input will have at least one output, no matter how many violations it incurs.[3]

In order to be able to give preference to output forms that incur the fewest violations of a constraint C, we first mark the violations and then select the best candidates using lenient composition. We set a limit n, an upper bound for the number of violations that the system will consider, and employ a set of auxiliary constraints, $\mathtt{V}_{n-1}$, $\mathtt{V}_{n-2}$, ..., $\mathtt{V}_0$, where $\mathtt{V}_i$ accepts the output candidates that violate the constraint at most i times. The most stringent enforcer, $\mathtt{V}_0$, allows no violations. Given a relation R, a mapping from the inputs to the current set of output candidates, we mark all the violations of C and then prune the resulting R' with lenient composition: R' .O. $\mathtt{V}_{n-1}$ .O. $\mathtt{V}_{n-2}$ ... .O. $\mathtt{V}_0$. If an input form has output candidates that are accepted by $\mathtt{V}_i$, where $n > i \geq 0$, all the ones that are rejected by $\mathtt{V}_i$ are eliminated; otherwise the set of output candidates is not reduced. The details of this strategy are explained in section 1.4.2.

For the sake of efficiency, we may compose all the inputs with the GEN relation and leniently compose the result with all the constraints into a single finite-state transducer that maps each input form directly into its optimal surface realizations, and vice versa.

1.4 Finite-State OT Prosody

In this section, we will show in as much detail as space allows how the two OT descriptions of Finnish prosody in Section 1.2 can be implemented in a finite-state system. The regular expression formalism in this section and the **xfst** application used for computation are described in the book *Finite State Morphology* (Beesley and Karttunen, 2003). This technology is the result of a long line of research started by Ronald M. Kaplan and Martin Kay in the early 1980s.

1.4.1 The GEN Function

The task of the GEN function is to provide each input with all conceivable output candidates. In keeping with the hallmark OT thesis of "freedom of analysis", every candidate, however bizarre, should be available for evaluation by the constraints.

A GEN function for prosody must accomplish three tasks: (1) parse the input into syllables, (2) assign optional stress, and (3) combine syllables optionally into metrical feet. Each of these tasks can be performed by a finite-state transducer. The GEN function for Finnish prosody can thus be defined as the composition of the three compo-

[3] Frank and Satta (1998, pp. 8–9) call this operation "conditional intersection."

nents:[4] `Syllabify .o. Stress .o. Scan`, where `.o.` represents ordinary composition, as opposed to `.O.` for lenient composition. With the help of this regular expression, we can define the GEN function for Finnish prosody as in (8)

(8) `define GEN(X) [X .o. Syllabify .o. Stress .o. Scan]`

where `X` can be a single input form or a symbol representing a set of input forms or an entire language. The result of evaluating `GEN(X)` is a transducer that maps each input form in `X` into all of its possible output forms.

The initial task, syllabification, is non-trivial in Finnish because the nucleus of a syllable may consist of a short vowel, a long vowel, or a diphthong. Adjacent vowels that cannot constitute a diphthong must be separated by a syllable boundary. For example, the first vowel pair in the input *kielien* 'tongue' (Pl. Gen.) constitutes a diphthong but the second pair does not because of its position in the word. The correct syllabification is kie.li.en.[5]

Because stress assignment and foot assembly are optional, GEN produces a large number of alternative prosodic structures for even short inputs. For example, for the input *kala* 'fish' (Sg. Nom.), `GEN({kala})` produces the 33 output forms shown in (9).

(9) kà.là, kà.lá, kà.la, kà.(lá), kà.(là), ká.là, ká.lá, ká.la, ká.(lá), ká.(là), ka.là, ka.lá, ka.la, ka.(lá), ka.(là), (ká).là, (ká).lá, (ká).la, (ká).(lá), (ká).(là), **(ká.la)**, (ká.lá), (ká.là), (kà).là, (kà).lá, (kà).la, (kà).(lá), (kà).(là), (kà.la), (kà.lá), (kà.là), (ka.lá), (ka.là)

As the analyses by Elenbaas and Kiparsky predict, the correct output is (ká.la).

1.4.2 The Constraints

There are two types of violable OT constraints. For CATEGORICAL constraints, the penalty is the same no matter where the violation occurs. For GRADIENT constraints, the site of violation matters. For example, All-Feet-Left assigns to non-initial feet a penalty that increases with the distance from the beginning of the word.[6]

[4] For details, see the **xfst** script in `http://www.stanford.edu/~laurik/fsmbook/examples/FinnishOTProsody.html`.

[5] Instead of providing the syllabification directly as part of GEN, it would of course be possible to generate a set of possible syllabification candidates from which the winners would emerge through an interaction with OT constraints such as HaveOnset, FillNucleus, NoCoda, etc.

[6] The current status of gradient constraints is controversial. McCarthy (2003) argues that gradient constraints are unnecessary and harmful. According to him, alignment constraints such as (4i) should be categorical. See also Eisner (2000).

Our general strategy is as follows. We first define an evaluation template for the two constraint types and then define the constraints themselves with the help of the templates. We use asterisks as violation marks and use lenient composition to select the output candidates with the fewest violation marks. Categorical constraints mark each violation with an asterisk. Gradient constraints mark violations with sequences of asterisks starting from one and increasing with the distance from the word edge.

The initial set of output candidates is obtained by composing the input with GEN. As the constraints are evaluated in the order of their ranking, the number of output forms is successively reduced. At the end of the evaluation, each input form typically should have just one correct output form.

An evaluation template for categorical constraints, shown in (10), needs four arguments: the current output mapping, a regular expression pattern describing what counts as a violation, a left context, and a right context.[7]

(10)
```
define Cat(Candidates, Violation, Left, Right) [
Candidates .o. Violation -> ... "*" || Left _ Right
.O. Viol3 .O. Viol2 .O. Viol1 .O. Viol0
.o. Pardon ];
```

The first part of the definition composes the candidate set with a rule transducer that inserts an asterisk whenever it sees a violation that occurs in the specified context.[8] The second part of the definition is a sequence of lenient compositions. The first one eliminates all candidates with more than three violations, provided that some candidates have only three or fewer violations. Finally, we try to eliminate all candidates with even one violation. This will succeed only if there are some output strings with no asterisks. The auxiliary terms `Viol3`, `Viol2`, `Viol1`, `Viol0` limit the number of asterisks. For example, `Viol1`, is defined as `~[$"*"]^2`. It prohibits having two or more violation marks. The third part, `Pardon`, is defined as `"*" -> 0`. It removes any remaining violation marks from the output strings. Because we are counting violations only up to three, we cannot distinguish strings that have four violations from strings with more than four violations. It turns out that three is an empirically sufficient limit for our categorical prosody constraints.

The evaluation template for gradient constraints counts up to 14 violations and each violation incurs more and more asterisks as we

[7] Some constraints can be specified without referring to a particular left or right context. The expression `?*` stands for any unspecified context.

[8] The formalism is explained in Chapter 2 of Beesley and Karttunen (2003).

count instances of the left context. The definition is given in (11).

(11)
```
define GradLeft(Candidates, Violation, Left, Right) [
Candidates
.o. Violation -> "*" ... ||.#. Left _ Right
.o. Violation -> "*"^2 ... ||.#. Left^2 _ Right
.o. Violation -> "*"^3 ... ||.#. Left^3 _ Right
.o. Violation -> "*"^4 ... ||.#. Left^4 _ Right
.o. Violation -> "*"^5 ... ||.#. Left^5 _ Right
.o. Violation -> "*"^6 ... ||.#. Left^6 _ Right
.o. Violation -> "*"^7 ... ||.#. Left^7 _ Right
.o. Violation -> "*"^8 ... ||.#. Left^8 _ Right
.o. Violation -> "*"^9 ... ||.#. Left^9 _ Right
.o. Violation -> "*"^10 ... ||.#. Left^10 _ Right
.o. Violation -> "*"^11 ... ||.#. Left^11 _ Right
.o. Violation -> "*"^12... || .#. Left^12 _ Right
.o. Violation -> "*"^13 ... ||.#. Left^13 _ Right
.o. Violation -> "*"^14 ... ||.#. Left^14 _ Right
.O. Viol14 .O. Viol13 .O. Viol12 .O.Viol11 .O. Viol10
.O. Viol9 .O. Viol8 .O. Viol7 .O. Viol6 .O. Viol5
.O. Viol4 .O. Viol3 .O. Viol2 .O. Viol1 .O.  Viol0
.o. Pardon ];
```

Using the two templates in (10) and (11), we can now give very simple definitions for Kiparsky's nine constraints in (4). We only need a few auxiliary concepts listed in (12). We omit the simple definitions here.

(12)
a. `Light`: Light Syllable (A syllable with a short vowel and without a coda)
b. `MSS`: Syllable with Main Stress
c. `SV`: Stressed Vowel
d. `SS`: Stressed Syllable
e. `US`: Unstressed Syllable
f. `S`: Syllable
g. `E`: Edge: Syllable Boundary or Word Edge.
h. `B`: Boundary (Edge or Foot Boundary)

The constraints are defined in (13).

(13)
a. *Clash: No stress on adjacent syllables.
`define Clash(X) Cat(X, SS, SS B, ?*);`
b. Left-handedness: The stressed syllable is initial in the foot.
`define AlignLeft(X) Cat(X, SS, ".", ?*);`

c. Main Stress: The primary stress in Finnish is on the first syllable.
```
define MainStress(X)
  Cat(X, ∼[B MSS ∼$MSS], .#., .#.);
```

d. Foot-Bin: Feet are minimally bimoraic and maximally bisyllabic.
```
define FootBin(X)
  Cat(X, "(" Light ")" | "(" S ["." S]^>1, ?*, ?*);
```

e. Lapse: Every unstressed syllable must be adjacent to a stressed syllable or to the word edge.
```
define Lapse(X) Cat(X, US, [B US B], [B US B]);
```

f. Non-Final: The final syllable is not stressed.
```
define NonFinal(X) Cat(X, SS, ?*, ∼$S .#.);
```

g. Stress-To-Weight: Stressed syllables are heavy.
```
define StressToWeight(X) Cat(X, SS & Light, ?*, B);
```

h. License-σ: Syllables are parsed into feet.
```
define Parse(X) Cat(X, S, E, E);
```

i. All-Ft-Left: The left edge of every foot coincides with the left edge of some prosodic word.
```
define AllFeetFirst(X)
  GradLeft(X, "(", ∼$"." "." ∼$".", ?*);
```

To take just one example, let us consider the `StressToWeight` function. The violation part of the definition, `SS & Light`, picks out syllables such as *tí* and *tì* that are light and contain a stressed vowel. The left context is irrelevant, represented as `?*`. The right context matters. It must be some kind of boundary; otherwise perfectly well-formed outputs such as (má.te).ma.(tìik.ka) would get two violation marks: (má*.te).ma.(tì*ik.ka). That is because *tì* by itself is a stressed light syllable but *tìik* is not. The violation mark on the initial syllable *má* is correct but has no consequence because the higher-ranked `MainStress` constraint has removed all competing output candidates for *matematiikka* 'mathematics' (Sg. Nom.) that started with a secondary stress, *mà*, or without any stress, *ma*.

1.4.3 Combining GEN with the Constraints

Having defined both the GEN function and Kiparsky's nine prosody constraints, we can now put it all together creating a single function, `FinnishProsody`, that should map any Finnish input into its correct prosodic form. The definition is given in (14).

(14)
```
define FinnishProsody(Input) [ AllFeetFirst( Parse(
StressToWeight( NonFinal( Lapse( FootBin( MainStress(
AlignLeft( Clash( GEN( Input ))))))))) ];
```

A regular expression of the form `FinnishProsody(X)` is computed "inside-out." First the GEN function defined in (8) maps each of the input forms in `X` into all of its possible output forms. Then the constraints defined in Section 1.4.2 are applied in the order of their ranking to eliminate violators, making sure that at least one output form remains for all the inputs that have at least one output form that does not run afoul of some unviolable constraint.

Applying `FinnishProsody` to the input *opettamassa* 'teaching' (Sg. Ine.) results in the input/output relation shown in (15) that correctly represents the ternary prosodic pattern of the word. The incorrect trochaic output competitor, (ó.pet).(tà.mas).sa, has been eliminated.

(15)
```
  o   p e t    t a    m a s   s a
( ó . p e t ) . t a . ( m à s . s a )
```

Getting the right result for one input is of course no guarantee that all possible inputs get the desired output. To provide a quick test for the correctness of the analysis we defined `FinnWords` as the set of 25 input words collected from Kiparsky and Elenbaas illustrating various patterns of light and heavy syllables. It is by no means a complete inventory, but it is sufficient to reveal that both analyses are flawed. The compilation of the regular expression `FinnishProsody(FinnWords)` with Kiparsky's constraints produces the output forms shown in (16).

(16) (ér.go).(nò.mi).a, (íl.moit).(tàu.tu).mi.(sès.ta), (íl.moit).(tàu.tu).(mì.nen), (ón.nit).(tè.le).(mà.ni).kin, (ó.pis).(kè.li).ja, (ó.pet).ta.(màs.sa), (vói.mis).te.(lùt.te).le.(màs.ta), (strúk.tu).ra.(lìs.mi), (rá.vin).(tò.lat), (rá.kas).ta.(jàt.ta).(rè.na).ko, (ré.pe).(ä̀.mä), (pé.ri).jä, (pú.he).li.(mèl.la).ni, (pú.he).li.(mìs.ta).ni, (mä́.ki), (má.te).ma.(tìik.ka), (mér.ko).(nò.min), (kái.nos).(tè.li).jat, (ká.las).te.(lèm.me), **(ká.las).te.(lè.mi).nen**, (ká.las).(tè.let), (kú.nin).gas, **(jä̆r.jes).tel.(mä̀l.li).syy.(dèl.lä).ni**, (jä̆r.jes).(tèl.mät).tö.(mỳy.des).(tä̀n.sä), (jä̆r.jes).(tèl.mäl).(lìs.tä).mä.(tö̀n.tä)

For a native speaker of Finnish, it is immediately obvious that the two bold-faced outputs in (16) are incorrect. The correct outputs are (ká.las).(tè.le).(mì.nen) and (jä̆r.jes).(tèl.mäl).li.(sỳy.del).(lä̀.ni). Why are the rightful winners losing to undeserving competitors?

1.5 Error Analysis

The GEN function produces 70,653 output candidates for *kalastelеminen* 'fishing' (Nom. Sg.). The six most highly ranked constraints, `Clash`, `AlignLeft`, `MainStress`, `FootBin`, `Lapse` and `NonFinal`, eliminate nearly all of them, leaving just two candidates to be evaluated by the next constraint, `StressToWeight`: (ká.las).te.(lè.mi).nen and (ká.las).(tè.le).(mì.nen). As shown in (17), the desired winner, (17b), has one `StressToWeight` violation more than its competitor (17a).

(17) a. (ká*.las).te.(lè*.mi).nen
b. (ká*.las).(tè*.le).(mì*.nen)

Consequently, the incorrect (17a) is left as the sole survivor.

The same problem arises in the case of *järjestelmällisyydelläni* 'with my systematicity' (Sg. Ade.). After the six most highly ranked constraints have been applied, out of the initial set of 21,767,579 output candidates, 36 candidates are left. As shown in (18), the desired winner, (18b), contains one `StressToWeight` violation whereas 8 others, including the actual winner (18a), satisfy the `StressToWeight` constraint.

(18) a. (jä̀r.jes).tel.(mä̀l.li).syy.(dèl.lä).ni
b. (jär.jes).(tèl.mäl).li.(sỳy.del).(lä̀*.ni)

In these two cases, the desired result could be obtained by switching the ranking of `Parse` and `StressToWeight`. However, the new ranking would have undesirable consequences elsewhere. In particular, it would produce the wrong result in the case of *rakastajattarenako* 'mistress' (Sg. Ess., QP) and *voimisteluttelemasta* 'having someone do gymnastics' (Sg. Ela.). Instead of the correct result shown in (16), they would surface with an unwanted trochaic pattern as in (19).

(19) a. *(rá.kas).(tà.jat).(tà.re).(nà.ko)
b. *(vói.mis).(tè.lut).(tè.le).(màs.ta)

Replacing Kiparsky's STRESS-TO-WEIGHT by the more specific *(ÙH) constraint proposed in Elenbaas (1999) and in Elenbaas and Kager (1999) yields the correct result in the case of *järjestelmällisyydelläni* because (lä̀.ni) in the last foot of (18b) is not a violation of *(ÙH). However, this switch does not help in the case of *kalastelеminen*. Instead of (17), we now get (20). The correct output, (20b), is still eliminated because it has one violation more than its competitor.

(20) a. (ká*.las).te.(lè.mi).nen
b. (ká*.las).(tè.le).(mì*.nen)

The specter of an unexpected competitor suddenly emerging to eliminate the desired winner is the bane of OT analyses.

The appendix of Elenbaas (1999) contains an extensive list of Finnish syllable patterns and a sample output or outputs for each pattern, e.g. XXLHLL (má.te).ma.(tìik.ka).ni. Here L and H stand for light and heavy syllable, respectively, and X can be either L or H. Although the list appears complete, there are gaps. The missing patterns include at least four where the Elenbaas analysis in fact gives an incorrect result: XXLLLH *(ká.las).te.(lè.mi).nen, XXHHLH *(há.pa).roi.(tùt.ta).vaa, XXHHLH *(pú.hu).(tè.tuim).(mìs.ta).kin and XXHLLH *(kú.ti).tet.(tù.ja).kin. Kiparsky's analysis also fails with the XXLLLH, XXHHLH and XXHHLH patterns but succeeds with (kú.ti).(tèt.tu).(jà.kin).

In the case of the XXHHLH pattern, the input *haparoituttavaa* 'of the kind that causes one to fumble' (Sg. Par.) has four remaining output candidates at the point where Kiparsky's Stress-To-Weight and Elenbaas' *(ÌLH) constraints come into play. As shown in (21), the desired winner, (21d), loses to (21a) and (21b), which have no violations.

(21) a. (há.pa).roi.(tùt).ta.vaa
b. (há.pa).roi.(tùt.ta).vaa
c. (há.pa).(roi).tut.(tà*.vaa)
d. (há.pa).(ròi.tut).(tà*.vaa)

At the next step, the contest between the remaining two incorrect outputs, (21a) and (21b), is decided in favor of (21b) because it has fewer violations of the `Parse` constraint (Kiparsky's License-σ, Elenbaas' Parse-Syl) than (21a). As in the case of (17) and (18), giving `Parse` a higher rank would give us the right result for the XXHHLH pattern. But, as we have already seen in (19), it would lead to errors elsewhere.

In the case of the XXLHHLH input *puhutetuimmistakin* 'even those who have been made to talk the most' (Pl. Ela.), the desired winner, (22b), loses at the end because it has more violations of the All-Ft-Left constraint than its only remaining competitor, (22a).

(22) a. (pú.hu).**(tè.tuim).****(mìs.ta).kin
b. (pú.hu).te.***(tùim.mis).*****(tà.kin)

In the case of the XXHLLH input *kutitettujakin* 'even the ones that have been tickled' (Pl. Par.), the expected winner, (23c), is eliminated under the Elenbaas analysis because it violates the *(ÌLH) constraint, whereas one of its two competitors, (23a), has no *(ÌLH) violation.

(23) a. (kú.ti).tet.(tù.ja).kin
b. (kú.ti).(tèt).tu.(jà*.kin)
c. (kú.ti).(tèt.tu).(jà*.kin)

Kiparsky's analysis does better in this case because the (tù.ja) foot in (23a) is a violation of his more general Stress-To-Weight constraint. Thus all three candidates in (23) tie on Stress-To-Weight, and License-σ gets to pick (23c) as the rightful winner.

As far as we can see, there is no ranking of the nine constraints in (4) that would produce the right outcome in all the cases we have discussed. Replacing Stress-To-Weight by *(ÙH) does not help.

1.6 Conclusion

The assumption that is common to Kiparsky, Elenbaas and Kager is that the alternation between binary and ternary patterns in Finnish arises in a natural way from the interaction of independently motivated universal constraints. The ranking of the constraints can presumably be discovered by examining a limited set of examples. It may ultimately turn out to be the right assumption for some set of constraints. But it is not true for the constraints that have been proposed so far.

Optimality Prosody is a difficult enterprise. There are computational tools such as **OTSoft**[9] and **Praat**[10] that can select the most optimal output candidate provided that the user has explicitly specified the competing output forms and has manually marked up the violations.[11] These tools presuppose that (1) all the relevant input types are covered and (2) all the possible output candidates are included. Neither condition is met in the Elenbaas and Kiparsky studies. Without a GEN function to enumerate all the possible outputs, it is easy to miss the actual winner even if one is a native speaker of the language and an expert in the field.[12]

A finite-state approximation of OT guards against some errors made by a human GEN and EVAL. Instead of working with individual words one-by-one, the phonologist can collect a set of possible inputs of all the different types and apply the constraint system to the whole corpus at once to see the global effect of any change. But even with the **xfst** techniques described in this paper, debugging OT constraints is a very hard problem.

Acknowledgments

Thanks to Arto Tapani Anttila, Kenneth R. Beesley, Mary Dalrymple, Tracy King, Paul Kiparsky, Annie Zaenen and an anonymous reviewer for their helpful comments on earlier versions of this paper.

[9] http://www.linguistics.ucla.edu/people/hayes/otsoft/

[10] http://www.fon.hum.uva.nl/praat/

[11] OTSoft can mark some types of simple violations automatically but not others.

[12] Quandoque bonus dormitat Homerus.

References

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. Stanford, CA: CSLI Publications.

Eisner, Jason. 2000. Directional constraint evaluation in Optimality Theory. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 257–263. Saarbrücken, Germany.

Elenbaas, Nine. 1999. *A Unified Account of Binary and Ternary Stress*. Utrecht, Netherlands: Graduate School of Linguistics.

Elenbaas, Nine and René Kager. 1999. Ternary rhythm and the lapse constraint. *Phonology* 16:273–329.

Frank, Robert and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24(2):307–316.

Hanson, Kristin and Paul Kiparsky. 1996. A theory of metrical choice. *Language* 72:287–436.

Kager, René. 1999. *Optimality Theory*. Cambridge, England: Cambridge University Press.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20(3):331–378.

Kaplan, Ronald M. and Paula S. Newman. 1997. Lexical resource reconciliation in the Xerox Linguistic Environment. In *ACL/EACL'98 Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pages 54–61. Madrid, Spain.

Karttunen, Lauri. 1998. The proper treatment of optimality in computational phonology. In *FSMNLP'98.*. Ankara, Turkey: Bilkent University. cmp-lg/9804002.

Kiparsky, Paul. 2003. Finnish noun inflection. In D. Nelson and S. Manninen, eds., *Generative Approaches to Finnic and Saami Linguistics: Case, Features and Constraints*, pages 109–161. Stanford, California: CSLI Publications.

Koskenniemi, Kimmo. 1983. Two-level morphology. Publication 11, University of Helsinki, Department of General Linguistics, Helsinki.

McCarthy, John J. 2002. *The Foundations of Optimality Theory*. Cambridge, England: Cambridge University Press.

McCarthy, John J. 2003. OT constraints are categorical. *Phonology* 20(1):75–138.

Prince, Alan and Paul Smolensky. 1993. *Optimality Theory: Constraint Interaction in Generative Grammar*. Rutgers, New Jersey: Cognitive Science Center. ROA Version, 8/2002.