Explorations on Positionwise Flag Diacritics in Finite-State Morphology

Anssi Yli-Jyrä
Department of Modern Languages, PO Box 3, 00014 University of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

Abstract

A novel technique of adding *positionwise flags* to one-level finite state lexicons is presented. The proposed flags are kinds of morphophonemic markers and they constitute a flexible method for describing morphophonological processes with a formalism that is tightly coupled with lexical entries and rule-like regular expressions. The formalism is inspired by the techniques used in two-level rule compilation and it practically compiles all the rules in parallel, but in an efficient way. The technique handles morphophonological processes without a separate morphophonemic representation. The occurrences of the allomorphophonemes in latent phonological strings are tracked through a dynamic data structure into which the most prominent (i.e. the best ranked) flags are collected. The application of the technique is suspected to give advantages when describing the morphology of Bantu languages and dialects.

1 Introduction

Computational morphology continues to be an area of challenges when it comes to the construction of exact models for many under-resourced languages whose grammar is not fully known. The purpose of this paper is to introduce a new flexible technique to finite state morphology: *positionwise flag diacritics* (or shortly: *positionwise flags, p-flags*). P-flags can be combined with the usual *flag diacritics* (Beesley and Karttunen, 2003) that are based on left-to-right inheritance. They are, however, directionless and relate the paths at a position rather than the positions on a path. As such, they capture phenomena for which morphophonological rules are often preferred to the usual flag diacritics.

The description of morphophonological processes is often delegated to SPE-inspired rewriting rules that are implemented as a sequence of finite transducers (Beesley and Karttunen, 2003). Such rules are particularly handy when describing suppletion or regular alternations that span over several segments.

Two-level systems (Koskenniemi, 1983) consist of parallel rules while being equally capable with transducer sequences. Since these systems have only one underlying representation, the ordering problems characterizing the transducer cascades are now avoided. However, over-constrained two-level systems are often difficult to debug or relax, especially when they involve multi segmental alternations and complex morphological conditions.

An ideal conflict resolution scheme is to ensure that each rule applies only in those string positions where the linguist wanted them to apply; if they apply more often, the risk for conflicting effects increases. In the typical implementations of two-level systems, the rules are separated from the description of the relevant morphemes. This makes it difficult to maintain coherence between the components of the system. There is thus a need for a better interface between the lexicon and the morphophonological rules.

An alternative approach avoids the morphophonological component by using the formalism of flag diacritics (Beesley and Karttunen, 2003). However, there remain morphophonological processes that are easier to describe with rules. We are thus posed with a question: *What is between the rules and flag diacritics*.

The main contribution of this article is the novel scheme of *positionwise flag diacritics* whose best practice involves capturing rule-like behaviour and elsewhere conditions. The second important contribution of the article is the idea of a normalized lexicon as a data structure that contains the latent phonological strings and their most prominent po-

Bolette Sandford Pedersen, Gunta Nešpore and Inguna Skadiņa (Eds.)
NODALIDA 2011 Conference Proceedings, pp. 262–269

sitionwise flags. Finally, the alleged relevance of the methods to computational description of *Bantu varieties* is made apparent.

1.1 The Prior Art

The flag diacritics described by Beesley and Karttunen (Beesley and Karttunen, 2003) implement an idea of expanding the state space of an automaton with a vector of flag value registers. During the construction of the networks, the flag diacritics are like ordinary multicharacter symbols. When a flag diacritic is encountered as a transition label during a lookup, it is interpreted (Table 1) and the symbol does not consume any input. A transition on a flag diacritic fails to exist if its success condition is not satisfied by the current flag value. Certain flag diacritics have side-effects on the flag values.

diacritic	succeeds if f is	side effect
`@P.`f`.`v`@`	*anything*	$f \leftarrow (v, +)$
`@N.`f`.`v`@`	*anything*	$f \leftarrow (v, -)$
`@C.`f`@`	*anything*	$f \leftarrow$ *undef*
`@U.`f`.`v`@`	*undef*, $(v, +)$ or $(\overline{v}, -)$	$f \leftarrow (v, +)$
`@D.`f`.`v`@`	*undef*, $(\overline{v}, +)$ or $(v, -)$	
`@D.`f`@`	*undef*	
`@R.`f`.`v`@`	$(v, +)$	
`@R.`f`@`	*anything but undef*	

Table 1: The conditions and side-effects of flag diacritics

The flag diacritics are designed the left-to-right scanning of the string in mind. The pattern (`@U.`f`.`v`@...@U.`f`.`v`@`) is typically used as a means to cope with effects that depend on a class, declension, conjugation, stem type alternation or a harmony. The patterns (`@P.`f`.`v`@...@R.`f`.`v`@`) and (...`@D.`f`.`v`@`) are typically used to check that a certain prefix is present and absent, respectively. The problem with this is that one ends up adding flag diacritics to such paths that *cannot* co-occur with the given prefix. Further mechanisms similar to the flag diacritics have been proposed in the literature (Blank, 1989; Blåberg, 1994; Kornai, 1996; Kiraz, 1997; Amtrup, 2004; Schmid, 2005; Trón et al., 2006; Cohen-Sygal and Wintner, 2006).

In the sequel, some prior knowledge of the two-level morphology (Koskenniemi, 1983) and the calculus of regular relations (Kaplan and Kay, 1994; Beesley and Karttunen, 2003) is assumed. Table 2 lists some crucial regular expression operators used in this paper. The rules used in two-level morphology provide a well understood basis for morphophonological grammars. In two-level rules, it is common to refer to a set of symbol pairs through an underspecific notation: e.g. :C refers to all correspondence pairs with a surface consonant.

operator	type	meaning
ϵ	nullary	empty string
:	infix	cross product
	infix	concatenation
*	postfix	Kleene star
$\cap$	infix	union
$\mid$ or $\cup$	infix	union
$-$	infix	asymmetric difference
$\circ$	infix	composition
$\Rightarrow$	infix	context restriction
$\Leftarrow$	infix	surface coercion
$\Leftrightarrow$	infix	the double arrow rule
Id()	function	convert to an identity relation

Table 2: regular expression operators

According to Koskenniemi and Silfverberg (2010), a state-of-the-art compilation method for morphophonological two-level rules is based on the *generalized restriction (GR)* operation (Yli-Jyrä and Koskenniemi, 2004). The operation is based on temporary markers that are added to a few positions of the strings during the construction. Besides its practical relevance to the compilation of individual rules, the GR operation provides theoretical means to compile and combine the set of disjunctively ordered constraints and the lexicon at the same time (Yli-Jyrä, 2008). Such a holistic approach relies on the determinization and complementation of finite automata. These operations are potential sources for high state and transition complexities.

2 Bantu Phonology

The Bantu languages refer to a group of a few hundred languages and dialects in Southern Africa. The languages share many common aspects such as rich affix morphology and reduplication. In phonology, the most common assimilatory process is the vowel height harmony of front and back vowels (Nurse and Philippson, 2003, p.46). The *Front Height Harmony (FHH)* process comes in two flavors that correspond to the rules such as follows:

(1) Front Height Harmony (FHH)
   a. General: /i/:[e] ⇔ [{o, e} C] _
   b. Extended: /i/:[e] ⇔ [{o, e, a} C] _.

The General FHH rule reads: "an underlying /i/ is realised as [e] if the vowel of the preceding syllable is [o] or [e]." The Extended FHH is more effective and it alters /i/ also after [a].

The General and Extended FHH rules are in conflict when /i/ is preceded with [a]. This kind of conflict is not to be resolved on the basis of rules per se, but by considering additional context information. The relevant context-dependency is neatly illustrated by the derivative extensions of verb stems in (Oshi)Kwanyama and (Otji)Herero varieties as described in (Halme, 2004) and (Möhlig and Kavari, 2008).

In Herero (HER), the inflectional stem of an applicative (APPL) verb (2-b) is formed by extending the inflectional stem (2-a) with the suffix /*-ir*/ that precedes the final vowel /*-a*/. In Kwanyama (KWA), the corresponding suffix is /*-il*/. In both cases, the sound /i/ is subject to the Extended FHH and the consonant is altered in nasal contexts (2-c).

(2) a. [*túng-a*] (HER)'build'-FV
   b. [*túng-ir-a*] (HER)'build'-APPL-FV
   c. [*món-en-á*] (KWA)'see'-APPL-FV

On the other hand, the Neuter verb (NEUT) is formed in both languages with the suffix /-ik/ (3-a). The /i/ sound in this suffix is subject to to General FHH in Kwanyama (3-b), but not in Herero (Möhlig and Kavari, 2008).

(3) a. [*túng-ik-á*] (KWA)'build'-NEUT-FV
   b. [*kómb-ek-á*] (KWA)'sweep'-NEUT-FV

The grammatical tone of the derivative extensions cannot be discussed here due to space constraints.

In sum, the Bantu phonology shows that the sound changes depend on the variety (HER~KWA), the morpheme position (root~verbal extension) and the grammatical category of a morpheme (APPL~NEUT). Such dependency can be described through the concept of morphophonemes.

3 Morphophonology

3.1 Morphophonemes by Variation

A technique for identifying morphophonemes has been given (Austerlitz, 1967). Another technique (Koskenniemi, 1991) would provide the Applicative and Neuter morphemes of Kwanyama with the following morphophonological structures where individual morphophonemes are wrapped in angle brackets (they must not be confused with the brackets that denote an orthographic string):

(4) a. |⟨i~e⟩⟨l~n⟩|
   b. |⟨i~e⟩⟨k⟩|

Both derivative extensions involve the same morphophoneme, |⟨i~e⟩|, which is distinguished from the morphophoneme |⟨i⟩| that occurs e.g. in verb roots. The morphophoneme helps thus to focus the application of the phonological rules on extensions (in contrast to verb roots).

The morphophoneme |⟨i~e⟩| is indifferent with the grammatical category of the extension. Therefore, the harmony rules need an additional condition to account the difference between APPL and NEUT. Reducing the morphological condition to the segmental structure of the morphemes does not seem to be a well-motivated option. Adding arbitrarily chosen lexical features (Koskenniemi, 1983, p.40) to the morphophonological strings is also a weakly motivated approach. Conditioning the rules with grammatical categories (Trost, 1991; Kiraz, 1997) is perhaps the most solid but also the heaviest approach. Conceptually, it leads to the following pair of realisation rules for the morphophoneme |⟨i~e⟩|:

(5) a. |⟨i~e⟩|:/i/ ⇒ _
   b. |⟨i~e⟩|:/e/ ⇔ /{e,a,o} C/ _(APPL),
   /{e,o} C/ _(NEUT).

The use of the grammar categories in rules is a complication that we want to avoid or make simpler.

3.2 Morphophonemes by Contexts

Recently, Koskenniemi and Silfverberg (2010) have shown that the multi context rules like (5-b) can be compiled as two separate rules by using the variants of the phonemes:

(6) a. |⟨i~e⟩|:/∥e_1∥
   ⇔ /∥{e_1,e_2,a,o} C∥ _(APPL)
   b. |⟨i~e⟩|:/∥e_2∥
   ⇔ /∥{e_1,e_2,o} C∥ _(NEUT).

The method requires that there is an additional mapping that replaces ∥e_1∥ and ∥e_2∥ with /e/.

It is worth observing that the multi context rules can be split in two different ways. By applying the

splitting to the underlying rather than the surface level, one captures the morphological distinction differentiated by the context conditions. This simplifies the contexts of the rules into more elegant rules:

(7) a. $|\langle$iE~eE$\rangle$|:/e/ $\Leftrightarrow$ /{e,a,o} C/ _
b. $|\langle$iG~eG$\rangle$|:/e/ $\Leftrightarrow$ /{e,o} C/ _.

In this case, there is no need for an additional mapping, because the refined morphophonemes can be used directly in the lexicon.

3.3 The GR-Based Compilation Method

A practical two-level grammar compilation method (Yli-Jyrä and Koskenniemi, 2006) reduces the traditional rules into *Generalized Restriction (GR)* rules. Accordingly, the double arrow rule (7-b) is first split into two subrules:

(8) $|\langle$iE~eE$\rangle$|:/e/ $\Rightarrow$ /{e,a,o} C/ _

(9) $|\langle$iE~eE$\rangle$|:/e/ $\Leftarrow$ /{e,a,o} C/ _.

The subrules are then reduced into two GR rules of the form $W \overset{2\diamond}{\Rightarrow} W'$. Each GR rule involves two argument languages, $W, W' \subseteq \Sigma^*\diamond\Sigma^*\diamond\Sigma^*$. For the moment, Σ is the set of feasible morphophoneme-phoneme pairs. The argument languages specify when the rule is triggered (W) and what the triggered rule then requires (W'). In the strings of these languages, the marker $\diamond$ is a string element indicating which morphophoneme-phoneme pair is in the focus.

In fact, the form of the GR rules can be simplified to $W \overset{1\diamond}{\Rightarrow} W'$ if we ignore certain epenthetic rules and are not interested in a length-based rule conflict resolution scheme that is supported by the full compilation method. In this case the argument languages W and W' are subsets of $\Sigma^*\diamond\Sigma^*$.

Under the simplification, the rules (8) and (9) give rise to the GR rules (10) and (11). In these rules, the *dots* symbol ... denotes the language Σ^*.

(10) $\ldots\diamond\langle$iE~eE$\rangle$:e... $\overset{1\diamond}{\Rightarrow}$
$\ldots\{$:e,:a,:o$\}$:C $\diamond\langle$iE~eE$\rangle$:e ...

(11) $\ldots\{$:e,:a,:o$\}$:C $\diamond\langle$iE~eE$\rangle$:... $\overset{1\diamond}{\Rightarrow}$
$\ldots\{$:e,:a,:o$\}$:C $\diamond\langle$iE~eE$\rangle$:e ....

One of the nice features of the GR-based approach is that the GR rules can be combined before they are compiled into finite automata over Σ. In other words, given the rules $W_1 \overset{2\diamond}{\Rightarrow} W_1'$ and $W_2 \overset{2\diamond}{\Rightarrow} W_2'$, there is an easy way to choose W_3 and W_3' in such a way that $W_3 \overset{2\diamond}{\Rightarrow} W_3'$ equals to the intersection of the first two rules.

Using this closure property of GR rules, we can represent the rules (7-a) and (7-b) with a single GR rule:

(12)
$$\left\{\begin{array}{c}\cdots\diamond_{\mathrm{E}}\langle\text{iE~eE}\rangle\text{:e}\ldots\\ \ldots\{\text{:e,:a,:o}\}\text{ :C}\diamond_{\mathrm{E}}\langle\text{iE~eE}\rangle\text{:}\ldots\\ \cdots\diamond_{\mathrm{G}}\langle\text{iG~eG}\rangle\text{:e}\ldots\\ \ldots\{\text{:e,:o}\}\text{ :C}\diamond_{\mathrm{G}}\langle\text{iG~eG}\rangle\text{:}\ldots\end{array}\right\}$$
$$\overset{1\{\diamond_{\mathrm{E}},\diamond_{\mathrm{G}}\}}{\Longrightarrow}$$
$$\left\{\begin{array}{l}\ldots\{\text{:e,:a,:o}\}\text{ :C}\diamond_{\mathrm{E}}\langle\text{iE~eE}\rangle\text{:e}\ldots\\ \ldots\{\text{:e,:o}\}\text{ :C}\diamond_{\mathrm{G}}\langle\text{iG~eG}\rangle\text{:e}\ldots\end{array}\right\}.$$

3.4 One-Level Morphology

There is a striking redundancy in the GR rule (12). Namely, the type of the harmony process, i.e. 'E' vs. 'G', is indicated in every string twice: by the morphophoneme and by the marker.

Due to this observation, there is a motivation to replace the marked morphophoneme-phoneme correspondence pair like $\diamond_{\mathrm{E}}\langle$iE~eE$\rangle$:e with a marked phoneme $\diamond_{\mathrm{E}}$e, reduce the two-level rules into *one-level rules* and redefine the alphabet Σ as a set of phoneme symbols (and some other symbols).

The key insight for valuing the resulting *one-level grammar* is that the morphophonemes and the markers specify the same underlying segmental positions. Roughly speaking, they are just two means to signal the associated variation and morphological contexts. The first is collective while the second is distributive. In order to switch fully to the one-level representation, we have to introduce a couple of useful ideas.

The *first idea* makes the one-level representation richer and sufficient for linguistically appropriate morphological analysis. In Bantu linguistics, the linguistically appropriate detail is often synonymous with phonemic transcriptions and the associated interlinear morpheme glosses. The format of the glossed linguistic examples is currently well regulated by the Leipzig Glossing Rules (Bickel et al., 2008).

A possible scheme for conflating the transcription and the glosses into a one-level string has been drafted by the author (Yli-Jyrä, 2011). This scheme would encode the example (2-c) as the following string where a white space separates individual symbols.

(13) `m ó n ::(KWA) :'see' - e n ::APPL - á ::FV`

The *second idea* is to open the GR rule in such a way that the markers in W and W' are more transparent and the lexicon can predetermine at least some of the marker positions. The markers can specify the positions of the morphophonemes and simultaneusly avoid postulating a single underlying string of morphophonemes. The goal of transparency is achieved by elaborating the notion of markers.

4 Positionwise Flags

Like the conventional flags (Beesley and Karttunen, 2003), the positionwise flag diacritics are designed for use in the lexicon. The positionwise flags are, however, in some sense more general because they have a further use in the morphophonological rule component. They correspond to the markers of the GR rule and can therefore be viewed as user's interface to the internals of the GR operation.

Let p be a name for a morphophonological *process* and k an even number of *ranks* in the positionwise flags. Based on these parameters, we define a set of markers and positionwise flags according to Table 3. Let P be the set of all names of morphophonological processes. For every such name, there is a series of ranked markers and flags. The p-flags and the corresponding markers are semantically equivalent, but used in different notations: The p-flags resemble the conventional flags and can be used in finite state lexicon formalisms whereas the corresponding markers are associated with the GR-based compilation formulas and its mathematical presentation. All ranked markers/p-flags constitute the set M.

marker	p-flag	feeded by	successful?
$\diamond_{1.\mathrm{p}}$	`@1.`p`@`		yes
$\diamond_{2.\mathrm{p}}$	`@2.`p`@`		no
$\diamond_{3.\mathrm{p}}$	`@3.`p`@`	rank 2	yes
$\diamond_{4.\mathrm{p}}$	`@4.`p`@`	rank 3	no
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\diamond_{\mathrm{k.p}}$	`@k.`p`@`	rank $k-1$	no

Table 3: Markers and positionwise flag diacritics

The integer number in each marker/flag specifies the respective rank. Previously, Frank Liang (Liang, 1983) have proposed ranked hyphenation patterns that lie at odd- and even-numbered levels. Following the same idea, an odd-ranked marker is used to indicating a position of a *successful* (or valid) allomorphophoneme while an even-ranked marker is used to represent a *failing* (or invalid) allomorphophoneme.

Beyond the rank 1, the markers that have a low rank *feed* the markers that have an immediately higher rank. This means that a higher marker has the power only if it has an overriding effect. In the context of hyphenation patterns (Liang, 1983), this is so obvious that one does not even think of it. In the current context, the principle of such feeding can greatly simplify the specification of contexts in the case of rule exceptions. Still, it does not lead to similar problems as the rule ordering of transducer cascades, because the feeding order correlates now with the disjunctive ordering of parallel rules rather than the multiple levels of phonological representation.

Explaining the meaning of the p-flags requires some related mathematical definitions. The function $h : (\Sigma \cup M)^* \to \Sigma^*$ is a string projection that essentially deletes the markers M in its input strings. The strings with a successful marker form the languae $S = \{x \diamond_{i.p} y \mid xy \in \Sigma^*, i \in \{1, 3, ..., k-1\}, p \in P\}$. The strings with an unsuccessful marker form the language $F = \{x \diamond_{i.p} y \mid xy \in \Sigma^*, i \in \{2, 4, ..., k\}, p \in P\}$. Finally, the strings with markers of rank 1 and 2 form the language $B = \{x \diamond_{i.p} y \mid xy \in \Sigma^*, i \in \{1, 1\}, p \in P\}$.

4.1 Flags and Markers in the Lexicon

An *uncompiled lexicon* L defines a subset of $(\Sigma \cup M)^*$. The format of this lexicon is such that strings may contain several markers. Such markings will be distributed in the *normalized lexicon* N_0 by reducing the number of markers in the string. The normalized lexicon is given by $N_0 = h(L) \cup \{h(x)\, m\, h(y) \mid m \in M, xmy \in L\}$ and it contains three components:

$$\boxed{N_0 \cap \Sigma^*} \quad \boxed{N_0 \cap F} \quad \boxed{N_0 \cap S}.$$

Define the relation $\mu : \Sigma^* M \Sigma^* \times \Sigma^* M \Sigma^*$ as the set $\{(x \diamond_{i.p} y, x \diamond_{j.p} y) \mid 1 \le j < i \le k, p \in P, x, y \in \Sigma^*\}$. Now the lexicon with positionwise flags is compiled into a set of unmarked strings by applying the GR rule as follows:

$$[N_0 \cap \Sigma^*] \cap [(N_0 \cap F) \overset{1M}{\Rightarrow} \mu(N_0 \cap S)].$$

By the definition of the GR rule (Yli-Jyrä and Koskenniemi, 2006), this reduces to

$$[N_0 \cap \Sigma^*] - h((N_0 \cap F) - \mu(N_0 \cap S)).$$

4.2 Flags and Markers in the Grammar

The normalized lexicon N_0 is accompanied by the *one-level grammar* $G \subseteq S \cup F$ that consists of marked strings. The set of marked strings is used in some sense like the more familiar notion of a rule set. Formally, this means that the grammar is itself a language. This language consists of two components:

$$\boxed{G \cap B} \quad \boxed{G - B}.$$

The first component is automatically activated for application to the strings, while the second component waits for joining the application when the lexicon or previously applied marked strings exhibit flags that could trigger further marked strings.

The application of the first component of the grammar gives rise to the second generation of the normalized lexicon:

$$N_1 = [N_0 \cup [G \cap B]] - \mu(N_0 \cup [G \cap B]).$$

The activation of the remaining marked strings in the grammar is based on the *increment* relation $\iota : \Sigma^* M \Sigma^* \times \Sigma^* M \Sigma^*$ defined as the set $\{(x \diamond_{i.p} y, x \diamond_{i+1.p} y) \mid 2 \leq i < k, p \in P, x, y \in \Sigma^*\}$. Through this, any further generation of the normalized lexicon is obtained as

$$N_{i+1} = [N_i \cup [G \cap \iota(N_i)]] - \mu(N_i \cup [G \cap \iota(N_i)]).$$

The number of ranks k determines how many generations of the normalized lexicon need to be computed. Because the second generation, N_1, can still introduce markers whose rank is 1, the last normalized lexicon to be computed is N_k. When this has been computed, we obtain the markerless language of (glossed) transcriptions by evaluationg the formula:

$$[N_k \cap \Sigma^*] - h((N_k \cap F) - \mu(N_k \cap S)).$$

4.3 The Lexicon-Grammar Teamwork

Due to the shared positionwise flags / markers, the lexicon and the grammar can help each other. The lexicon specifies the default allomorphophonemes and the grammar can implement the alternations that choose alternative allophonemes.

Consider the morphophoneme ⟨iE-eE⟩. In the one-level lexicon this morphophoneme becomes an ambiguity class $\{\diamond_{1,\mathrm{E}}\mathrm{i},\ \diamond_{2,\mathrm{E}}\mathrm{e}\ \}$. In a finite-state lexicon formalism, this ambiguity class could correspond to a user-defined constant symbol that denotes a regular expression `[@1.E@i | @2.E@e]`. The elements of the ambiguity class signal, on one hand, that the default allomorphophoneme is /i/ and that the allomorphophoneme /e/ is not acceptable by default. On the other, the markers/p-flags attached to the allomorphophonemes establish positions to which the grammar can refer. The grammar then alters the prominency of the marked allomorphophonemes by increasing their ranks through such marked strings that represent the appropriate environments of the non-default allomorphophonemes.

In practice, the grammar is given through some finite-state formalism that is now extended with the p-flag notation. In this formalism, the FHH rules are given through the regular expressions

(14)
```
...[e|o] C [@2.G@i|@3.G@e]...
...[e|a|o] C [@2.E@i|@3.E@e]...
```

whose union expresses the string set

$$(15) \qquad \left\{ \begin{matrix} \Sigma^*\{\mathrm{e,o}\}\mathrm{C}\{\diamond_{2.\mathrm{G}}\mathrm{i}, \diamond_{3.\mathrm{G}}\mathrm{e}\}\Sigma^* \\ \Sigma^*\{\mathrm{e,a,o}\}\mathrm{C}\{\diamond_{2.\mathrm{E}}\mathrm{i}, \diamond_{3.\mathrm{E}}\mathrm{e}\}\Sigma^* \end{matrix} \right\}.$$

The regular expressions should match full-fledged one-level representations, but the simplified expressions in (14) concern only the phonemic content. The first pattern in (14) states that if an allomorphophoneme /i/ governed by the General FHH process occurs after /e/ or /o/, it is wrong (rank 2 means a failure), but the allomorphophoneme /e/ in the same position is perfect (rank 3 means a success).

If one wishes, the regular expressions like (14) can be written with a more intuitive notation that mimics the rules of generative phonology but are actually born in the currently used one-level representation:

(16)
```
i2.G → e3.G / {e,o} C __
i2.E → e3.E / {e,a,o} C __.
```

5 Reflections and Further Work

It is noteworthy that the new grammar (14) is substantially simpler and more intuitive than the one in (12). It seems to express the linguistic generalizations such as (1) compactly and directly (but as

a morphophonological rather than a phonological process). These are just a few reasons that suggest that the positionwise flags may help increase the simultaneous human- and machine-readability of computational descriptions.

Another interesting thing is that the normalized lexicon is a dynamic data structure that contains latent phonological strings and their known allomorphophonemes. The resulting adaptability of the lexicon may have uses in machine learning and computational typology. The normalized lexicon is also a very efficient debugging tool if some rules are missing or too strict.

The current method does not support rules whose environment conditions refer to morphophonemic contexts. Further investigations are needed to find out how severe a restriction this is since sufficiently precise conditions are still often expressible through the surface string and the interspersed glosses. If necessary, the support for markers in the contexts can be added although this would complicate the current method. Another extension would allow the normalized lexicon to contain pairs of positions i.e. substrings too. Such a data structure could define autosegments such as tones and metrical structures and support feature spreading and other suprasegmental processes, for example. There is, however, a danger that such an extension would make the system intractable.

In order to use simpler GR rules in compilation, the current presentation ignored the epenthesis rules. It turns out that the one-level representation is closely related to the model of the Item and Arrangement morphology (Hockett, 1954). In the IA model, epenthesis is not a real problem.

The surface orientation of the system is not necessarily a bad thing from the methodological point of view. One-level rules are easier to write, especially when segments are governed by more than one morphophonological process. The processes have the minimum impact on each other's alphabet because the surface string and the glosses are not split into separate cases.

Further research can find specialized and more efficient algorithms for the manipulation of markers/p-flags. It is also possible to optimize the representation of the one-level grammar G. Rules whose context conditions contain gaps are currently expensive because they can have a big effect on the state complexity. The problem arises only in rules that have not been constrained with a lexicon.

The author has discovered that the finite-state representation of the grammar can be optimized through on-demand compilation techniques. For example, one should avoid the eager expansion of the dots symbol '...' when compiling the regular expressions in the grammar G. To compute the original semantics of the intersection $G \cap N_i$, one can compute the image of the regular relation $(\mathrm{Id}(G) \circ T_1) \circ (T_2 \circ \mathrm{Id}(N_i))$ where the intermediately composed relation T_1 substitutes the occurrences of the symbol ... with the language $\ldots^*$ and the relation T_2 substitutes the occurrences of the symbol ... with the alphabet Σ. Both compositions in the parentheses have only a small effect on the size of the automata and the final composition is then computed in a very efficient way. However, this approach works only if the regular expressions do not contain negations. Therefore, the author is looking for ways to generalize the optimization to regular expressions that contain negations.

Although the positionwise flags can be combined with the conventional flag diacritics, this might be an error-prone activity, at least without a careful design step during which the possible interactions are properly managed.

In the context of computational Bantu morphology (Bosch, 2010), conventional flag diacritics and morphophonological rules have been proven very useful. This may suggest that also the positionwise flags have relevance to the Bantu morphophonology and morphotactics.

The author has started to evaluate the applicability of the currently presented formalism in the context of the computational description of tonal Bantu languages such as Kwanyama as well as in the construction of synchronic computational models of some closely related Bantu dialects. In this context, the one-level representation that reflects the Leipzig Glossing Rules (Bickel et al., 2008) is instrumental in fixing the desired morphosyntactic representation. The increased transparency of the grammar and the described strings is likely to support the notion of literate programming in the language documentation context.

References

Jan W. Amtrup. 2004. Efficient finite state unification morphology. In *Proceedings of the 20th international conference on Computational Linguistics*,

COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Austerlitz. 1967. The distributional identification of Finnish morphophonemes. *Language*, 43(1):20–33.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, CA, USA.

Balthasar Bickel, Bernard Comrie, and Martin Haspelmath. 2008. The Leipzig Glossing Rules. Conventions for interlinear morpheme by morpheme glosses. Max Planck Institute for Evolutionary Anthropology, Dept. of Linguistics. Retrieved 2011-02-15 from `http://www.eva.mpg.de/lingua/resources/glossing-rules.php`.

Olli Blåberg. 1994. *The Ment model. Complex states in finite-state morphology*. Number 27 in Reports from Uppsala University, Linguistics (RUUL). Department of Linguistics, Uppsala University, Uppsala, Sweden.

Glenn David Blank. 1989. A finite and real-time processor for natural language. *Communications of the ACM*, 32(10):1174–1189, October.

Sonja Bosch. 2010. Rule-based morphological analysis: Shared challenges, shared solution. In Karsten Legère and Christina Thornell, editors, *Bantu Languages: Analyses, Description and Theory*, volume 20 of *East African Languages and Dialects*, pages 1–15, Köln. Rüdiger Köppe Verlag.

Yael Cohen-Sygal and Shuly Wintner. 2006. Finite-state registered automata for non-concatenative morphology. *Computational Linguistics*, 32(1):49–82.

Riikka Halme. 2004. *A Tonal Grammar of Kwanyama*, volume 8 of *Namibian African Studies*. Rüdiger Köppe Verlag, Köln.

Charles Hockett. 1954. Two models of grammatical description. *Word*, 10:210–231.

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

George Anton Kiraz. 1997. Compiling regular formalisms with rule features into finite-state automata. In *35th ACL 1997, 8th EACL 1997, Proceedings of the Conference*, pages 329–336, Madrid, Spain.

András Kornai. 1996. Vectorized finite state automaton. In András Kornai, editor, *Extended Finite State Models of Language, Proceedings of the ECAI'96 Workshop*, Studies in Natural Language Processing, pages 36–41. Cambridge University Press.

Kimmo Koskenniemi and Miikka Silfverberg. 2010. A method for compiling two-level rules with multiple contexts. In *Proc. 11th ACL-SIGMORPHON, ACL 2010*, pages 38–45, Uppsala, Sweden, 15 July.

Kimmo Koskenniemi. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. Number 11 in Publications. Department of General Linguistics, University of Helsinki, Helsinki.

Kimmo Koskenniemi. 1991. A discovery procedure for two-level phonology. In A. Zampolli, L. Cignoni, and C. Peters, editors, *Computational Lexicology and Lexicography: A Special Issue Dedicated to Bernard Quemada*, volume I, pages 451–46. Giardini Editori e stampatori, Pisa.

Franklin Mark Liang. 1983. *Word Hy-phen-a-tion by Com-pu-ter*. Ph.D. thesis, Stanford University, Department of Computer Science, August.

Wilhelm J. G. Möhlig and Jekura U. Kavari. 2008. *Reference Grammar of Herero (Otjiherero)*. Number 3 in Southern African Languages and Cultures. Rüdiger Köppe Verlag, Köln.

Derek Nurse and Gérald Philippson. 2003. *The Bantu languages*. Routledge, New York.

Helmut Schmid. 2005. A programming language for finite state transducers. In Anssi Yli-Jyrä, Lauri Karttunen, and Juhani Karhumäki, editors, *FSMNLP*, volume 4002 of *LNCS/LNAI*, pages 308–309. Springer.

Viktor Trón, Péter Halácsy, Rebrus Peter, András Rung, Péter Vajda, and Simon Eszter. 2006. Hunmorph.hu: Hungarian lexical database and morphological grammar. In *Proceedings of LREC 2006*, pages 1670–1673.

Harald Trost. 1991. Recognition and generation of word form for natural language understanding systems: Integrating two-level morphology and feature unification. *Applied Artificial Intelligence*, 5(4):411–457.

Anssi Yli-Jyrä and Kimmo Koskenniemi. 2004. Compiling contextual restrictions on strings into finite-state automata. In Loek Cleophas and Bruce W. Watson, editors, *The Eindhoven FASTAR Days, Proceedings*, number 04/40 in Computer Science Reports, Eindhoven, The Netherlands. Technische Universiteit Eindhoven.

Anssi Yli-Jyrä and Kimmo Koskenniemi. 2006. Compiling generalized two-level rules and grammars. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *FinTAL*, volume 4139 of *LNCS*, pages 174–185. Springer.

Anssi Yli-Jyrä. 2008. Applications of diamonded double negation. In *Finite-State Methods and Natural Language Processing, 6th International Workshop, FSMNL-2007, Potsdam, Germany, September 14–16, Revised Papers*. Potsdam University Press, Potsdam.

Anssi Yli-Jyrä. 2011. Lifting interlinear morpheme glosses into plain strings in computational morphology. Manuscript 14p. Submitted in March 2011.

ISSN 1736-6305 Vol. 11
http://hdl.handle.net/10062/16955