

An OT-based Finite-State Implementation of Menominee Agreement Morphology

Jochen Trommer
Institute of Cognitive Science
University of Osnabrück
jtrommer@uos.de

Abstract

This paper describes a finite-state implementation of the complex agreement morphology in the Algonquian language Menominee. The analysis is based on Optimality Theory (Prince and Smolensky, 1993) and implemented in a computational system which realizes generation in OT-Morphology by extended finite-State techniques. Basic constraint types are defined and evaluated by modified versions of algorithms presented in Ellison (1994) and Karttunen (1998). Finite-State machines are created dynamically and evaluated according to the input.

1 Introduction

Because of their intricate agreement system, Algonquian languages have long been in the focus of theoretical work on inflectional morphology (e.g. Halle and Marantz, 1993). Menominee is an almost extinct Central Algonquian language documented in detail by Bloomfield (1962).

In this paper, I describe main features of an implementation of Menominee agreement morphology based on an account in Optimality Theory (OT, Prince and Smolensky, 1993; McCarthy and Prince, 1995). I will restrict myself to Menominee morphotactics which is very complex in itself and will not treat Menominee morphophonemics which is another complex area.

In section 2, I introduce some relevant morphological phenomena. In section 3, I present OT-constraints to account for these phenomena. Finally, in section 4, I discuss the algorithms of the finite-state implementation.

2 Basic Morphological Phenomena

2.1 Multiple Exponence

In many Menominee verb forms, single arguments are crossreferenced by more than one affix:¹

- (1) *po·se-w-ak*
embark-[+3]-[+pl]
'they embark' (p. 150)

While this is partially redundant in some cases, to distinguish forms, different affixes usually realize different aspects of an agreement trigger. Thus *-w* ([+3]) marks that the corresponding argument is 3rd person while *ak-* shows that it is plural. I will assume that multiple exponence of one argument by different affixes is motivated by the requirement to realize *all* features of the agreement trigger by affixes.

2.2 Infidelity

While Menominee has subject and object agreement, it uses the same affixes for both types of agreement.

¹Page numbers in the examples refer to Bloomfield (1962).

(2)

- a. *ke-na-n-a--w-a-w*
2-fetch-D-[+3]-[-1+pl]
'you (pl.) fetch him' (p. 153)
- b. *ke-na-n-eko-w-a-w*
2-fetch-D-[+3]-[-1+pl]
'he fetches you (pl.)' (p. 154)

Thus in (2a) the affixes *ke-... -a-w* mark 2pl subject while they mark 2pl object in (2b). Conversely *-w* marks a 3rd person object in a. and subject in b. The forms are only distinguished by the direction markers *-a·* and *-eko* (see below).

2.3 Hierarchy Based Competition

While Menominee is rather generous in assigning different agreement affixes to single arguments, it severely restricts cooccurrence of affixes for different arguments. Thus *-w* generally marks 3rd person arguments while *-m* marks non-third arguments, but if both would be licensed as in the forms in (2) only *-w* occurs. Similar constraints hold for most types of agreement affixes. Only one agreement prefix is allowed and also [+1 +pl] *-enaw* and [-1 +pl] *-a·w* exclude each other, where *-enaw* is always realized, leading to ambiguous forms:

- (3) *ke-na-tom-enene-m-enaw* (**-a·w*)
2-call-D-[-3]-pl
'we call you (sg. or pl.)' (pg. 157)

Thus the object in (3) is ambiguous between 2sg and 2pl because the plural marker *-a·w* which would make it unambiguous is suppressed in the presence of *-enaw*. I call the suppression of affixes in the presence of affixes of the same type "hierarchy-based competition" because in most analyses (including mine) it is assumed that agreement heads for subject and object "compete" for realization by a certain affix class and prominence hierarchies such as 1st > 3rd person determine which head "wins".

2.4 Direction Marking

The most notorious feature of Algonquian Inflection is "direction" marking in transitive verbs: If the subject is higher on a prominence hierarchy

than the object, a direct marker (*-a·* in (2a)) appears, if the object is higher an inverse marker appears (*-eko* in (2b)). The relevant hierarchy for Menominee hence contains the ranking 1st/2nd person > 3rd person, but is actually much complexer as is shown in (4):

$$(4) \quad \left\{ \begin{array}{l} [+2] \\ [+1] \end{array} \right\} > [-\text{spec}] > \text{prox} > \text{obv} > [-\text{an}]$$

prox(imate)/obv(iative) is a special distinction for all 3rd person arguments in Algonquian languages, where proximate corresponds roughly to topic-hood and obviative to non-topic-hood. [-spec] stands for an unspecified subject which induces passive-like semantics but is morphologically marked as a transitive subject. As is obvious from (2), direction marking is intimately related to hierarchy-based competition, and my analysis will actually treat it as special case of hierarchy-based competition.

2.5 Affix Order

Apart from mutual exclusiveness the content of affixes also determines the order of affixes which is roughly as follows:

- (5) Clitics > stem > Direct/Inverse > [+/-3] > [+/-1pl] > proximate pl/obviative

Interestingly, this corresponds closely to the order predicted for agreement affixes by the OT-account in Trommer (2003) where person generally precedes number agreement.

3 An OT-Account

The model of the grammar I assume is a non-conservative extension of the framework developed in Trommer (1999). The relevant constraints are described and justified on the basis of crosslinguistic evidence in Trommer (2002) and Trommer (2003).

With Halle and Marantz (1993) I assume that Morphology is an independent module of the grammar which interprets the output of syntax, where syntax operates on bundles of morphosyntactic features without phonological content. Thus for the form in (2a), syntax generates the string in (6):

(6) [+v]₁ [+2 +nom +pl]₂ [+3 +acc -pl]₃

The morphology component provides a lexicon *Lex* of vocabulary items (VIs). Each VI consists of a phonological string and a set of feature structures (FSs) each bearing a unique index and specifying morphosyntactic content. (7) lists some relevant lexical VIs for the input in (6). For lexical VIs indices are always zero, and therefore not represented here, they become only relevant for VIs which are used in word forms.

- (7) a. ke: [+cl+2]
 b. na·n: [+v]
 c. w: [+3]
 d. a·: [+Nom +an][+Acc]
 e. ek: [+Nom][+Acc +an]
 f. m: [-3]
 g. a·w: [-1+pl]₂

Based on *Lex*, morphology maps an input (such as (6)) to a string of VIs as in (8):

- (8) ke: [+cl+2]₂
 na·n: [+v]₁
 a·: [+Nom +an]₂[+Acc]₃
 w: [+3]₃
 wa·w: [-1+pl]₂

According to the principles of Optimality Theory (Prince and Smolensky, 1993), a grammar consist of a generator GEN, a function which maps an input to an infinite set of candidates, and a language-specific ranking of a set of universal constraints. I assume that the morphology module is just such a grammar enriched by a language-specific VI lexicon *Lex*. In the following subsections, I will lay out the characteristics of morphological GEN, and of the basic constraint types I assume. I will show how these constraints account for the phenomena described in section 2. Note that the indices of FSs in VIs allow two-level constraints as in Correspondence Theory (CT, McCarthy and Prince, 1995), but the coindexing mechanisms and the constraints referring to indices are formally different from the mechanisms in phonological CT to mirror the special properties of inflectional morphology.

3.1 GEN

I will call two VIs *index siblings*, if they are identical apart from the indices of their feature structures. A VI V is licensed in a language L by a syntactic input string $I = F_1 \dots F_n$ iff it is an index sibling to some VI in $Lex(L)$, and satisfies the following conditions:

1. For all indices i of FSs in V : $1 \leq i \leq n$
2. For all feature structures F'_j in V : $F'_j \sqsubseteq F_j$ (F_j subsumes the input FS with which it is coindexed)

Now GEN can be defined as follows:

- (9) **GEN** maps an Input I in a language L to the set of all strings consisting exclusively from VIs which are licensed by I in L .

These conditions ensure that only features from the input are present in the output VIs (hence no morphological constraint can enforce the appearance of additional features in the output) and that each feature structure in the output corresponds to exactly one head in the input (however a VI might have different FSs corresponding to different input FSs). Crucially, nothing excludes the possibility that more than one VI corresponds to an input FS. Hence this definition of GEN in principle allows the phenomenon of **multiple exponence** (section 2.1). Since FSs in VIs need not be identical to the corresponding input FSs, **infidelity** (section 2.2) is possible if FSs in VIs do not specify grammatical role or case ([+Nom] and [+Acc] in the examples).

3.2 Alignment

Alignment constraints require that a specified item is maximally close to a specified border. This is the standard constraint type in OT to account for **affix order**. Following Trommer (2003), I assume that morphological alignment is only sensitive to whether a VI is simple (contains only one feature structure) or complex (contains more than one feature structure), and to the content of its morphosyntactic feature structures, and further that all morphologic alignment constraints refer to the edges of the word form. Hence we get the format in (10)

- (10) **ALIGN(F, E, C)**: Count a constraint violation for each VI that intervenes between the designated edge E (*left* or *right*) of the word form and the most remote VI of complexity C (*simple* or *complex*) which contains a FS subsumed by F .

Since alignment constraints refer to absolute edges, I write them iconically as follows:

- (11) $[+pl]_{\text{simple}} \Leftrightarrow R$ ALIGN($[+pl]$,right, simple)
 $L \Leftrightarrow []_{\text{complex}}$ ALIGN($[]$,left, complex)

I usually omit the subscript "simple" since most constraints refer to simple VIs. (12) shows alignment constraints in action with example (1) *po-se-w-ak*, 'they embark', i.e., for the input $[+v]_1 [+Nom +3 +pl]_2$:

- (12) $po-se:[+v]_1 w:[+3]_2 ak:[+pl]_2$

	$[+v]$ L \Leftrightarrow	$[3]$ L \Leftrightarrow	$[+pl]$ \Leftrightarrow R
po-se-w-ak		*	
$po-se-ak-w$		**!	*
$w-po-se-ak$	*!		
$w-ak-po-se$	*!		*
$ak-po-se-w$	*!	**	**
$ak-w-po-se$	*!*	*	**

A crucial advantage of the alignment-based approach to affix order is that it allows to use crosslinguistically attested ordering principles to account for the specific affix order in single languages. Thus Trommer (2003) shows in detail that person crosslinguistically tends to align to the left word edge (reflected here in $L \Leftrightarrow [3]$) while number tends to align to the right edge (reflected by $[+pl] \Leftrightarrow R$).

3.3 COHERENCE

As alignment, coherence constraints are primarily motivated by their effects on affix order (see Trommer, 2002, for discussion). Intuitively they state for specific affix classes that "together should stand, what together belongs", i.e. affixes corresponding to the same syntactic head should not be interrupted by affixes belonging to another head. A definition of the constraint type effecting this is given in (13). Note that a VI is said to contain an

index i if it contains a FS with index i . A *matching VI* for the constraint is a VI of complexity C (*simple* or *complex*) that contains a FS subsumed by F .

- (13) **COHERENCE(F, C)**: Count a constraint violation for each matching VI V containing index i preceded by another matching VI V' containing index j such that $i \neq j$ and there is no matching VI V'' between V' and V .

Crucially for Menominee, the formulation in (13) has the side effect that highranked COHERENCE constraints block more than one affix of the same type. Thus COH(EREN)CE($[3]$,simple) disallows more than one affix simple VI specifying third person:²

- (14) $[+v]_1 [+2 +nom +pl]_2 [+3 +acc -pl]_3$

	COH($[3]$)
$-m [-3]_2 -w [+3]_3$	*!
$-m [-3]_2$	
$-w [+3]_3$	

Thus COHERENCE gives as one half of an account for **hierarchy-based competition**. It allows to capture the restriction to one argument for specific affix classes.

3.4 PARSE F

PARSE F captures the intuition that output VIs should realize as many input features as possible:

- (15) **PARSE F**: Count a constraint violation for each feature token F^I in a input FS FS^I which is not realized by a type-identical feature token F^O in an output FS FS^O coindexed with FS^I .

(16) shows how this works for the example in (1):

- (16) $po-se:[+v]_1 w:[+3]_2 ak:[+prox +pl]_2$

	PARSE F
$po-se-w-ak$	
$po-se-ak$	*!
$po-se-w$	*!*
$po-se$	*!***

²" $[3]$ " subsumes " $[+3]$ " and " $[-3]$ ".


3.5 Relativized PARSE

While PARSE F treats all features equal, **hierarchy-based competition** requires formal means to stipulate that features of more prominent arguments are realized in the context of less prominent ones. This is achieved by constraints which demand the realization of a feature in the context of a less prominent one in the input.

- (17) **PARSE F^{F^T/F^C}** : If the input has exactly one FS FS^1 such that $F, F^T \sqsubseteq FS^1$ and one FS FS^2 such that $F^C \sqsubseteq FS^2$, then count one constraint violation if F does not subsume a FS in the output coindexed with FS^1 . Otherwise count no violation.


Thus PRS $[3]^{[+an]/[-an]}$ requires that the feature [3] (i.e., [+3] or [-3]) of an animate ([+an]) argument should be realized if there is also an inanimate argument in the input. This captures the fact that m:[-3] "wins" in hierarchy-based competition over w:[+3] if the [+3] argument is [-an] (w:[+3] wins if it is [+an]). (18) shows how this guarantees a unique output for the example in (14) ([+/-an] added):

- (18) $[+v]_1 [2+an+nom+pl]_2 [3-an+acc-pl]_3$


	COH [3]	PRS [3] ^{[+an]/[-an]}
-m [-3] ₂ -w [+3] ₃	*!	
 -m [-3] ₂		
-w [+3] ₃		*!

Relativized PARSE constraints also account for **direction marking** given the VIs for direct (a:[+Nom +an][+Acc]) and inverse (ek:[+Nom][+Acc +an]) markers from (7). The following tableaux demonstrate the computation of the correct marker in the examples of (2):

- (19) Input: $[+Nom+2+an+pl]_1 [2+acc+3+an-pl]_2$

	PARSE [+an] ^{[+2]/[+3]}
 $[+Nom+an]_1 [2+acc]_2$	
$[+Nom]_1 [2+acc+an]_2$	*!

- (20) Input: $[+Nom+3+an-pl]_1 [2+acc+2+an+pl]_2$

	PARSE [+an] ^{[+2]/[+3]}
$[+Nom+an]_1 [2+acc]_2$	*!
 $[+Nom]_1 [2+acc+an]_2$	

Note the crucial importance of coindexing here: Both VIs specify the same features, but to satisfy the PARSE constraint [+an] must appear in a FS coindexed with the relevant input FS.

4 The Finite-State Implementation

The implementation covers the complete verb paradigms for 4 basic verb types (intransitive verbs with animate/inanimate subject and transitive verbs with animate/inanimate objects).

It is designed to test the OT-analysis presented in section 3 for correctness. While many aspects of Menominee inflection (such as affix order) could be implemented in a more efficient way by finite-state machines (FSMs), this is dismissed to model closely the linguistically motivated analysis. The implementation is based on my own C implementation of basic finite-state machines and operations such as intersection union and the algorithms in Ellison (1994) and Karttunen (1998). As an alphabet for the FSMs I use finite sets of VIs usually augmented by some atomic symbol such as "*" (to represent constraint violations). Arcs in FSMs are labeled by single symbols (usually VIs) or by predicates conjoined by logical operators (cf. Eisner, 1997). In departure from the cited literature I use output structures which are coindexed with the input FSs integrating an adapted version of correspondence theory (McCarthy and Prince, 1995).

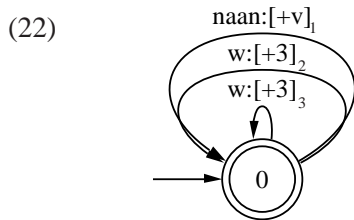
Since some constraints refer to indices to capture input-output relations, their application uses dynamically created FSMs. In other words to apply a constraint a new FSM is created according to the respective input and then used to compute the new candidate set. Thus constraints are implemented as filters which map regular sets and inputs (i.e., strings of FSs) to regular sets while GEN takes inputs and the VI lexicon *Lex* and maps it to the candidate set in form of a FSA. Thus a con-

straint ranking $F_1 \dots F_n$ and an input I are evaluated as follows:

$$(21) F_n(I, \dots F_2(I, F_1(\text{Gen}(I, Lex))))$$

4.1 GEN

Since the VI lexicon $Lex(L)$ and the set of indices in a given input I are finite, a simple function $enumerate(I, L)$ returns the set of VIs licensed by I in L . Now GEN is simply the Kleene closure for $enumerate(I, L)$. For example, the input $[+v]_1 [+Nom +3]_2 [+Acc +3]_3$ along with the lexicon in (7) results in the following automaton:

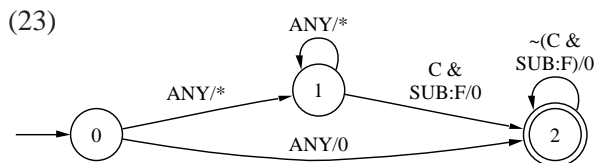


Note that some VIs from the lexicon are not represented in the automaton since they are not licensed by the input.

4.2 Alignment

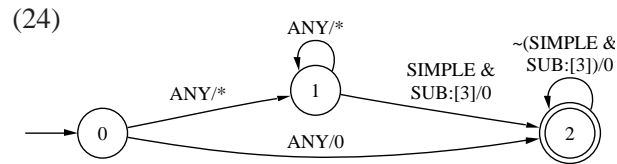
Notice that my version of alignment is not equivalent to Generalized Alignment (McCarthy and Prince, 1993) which can not be captured by finite-state devices since it refers to potentially multiple edges and constraint targets (Eisner, 1997). In the formulation of alignment in 3.2, each constraint has a unique edge (the left or right word edge) and a unique target to be aligned (the most remote item identified by the constraint) and *can* be captured by FSMs.

A left alignment constraint $\text{ALIGN}(F, \text{left}, C)$ is implemented by a finite state transducer of the form in (23), where "*" denotes one and "0" no constraint violation:

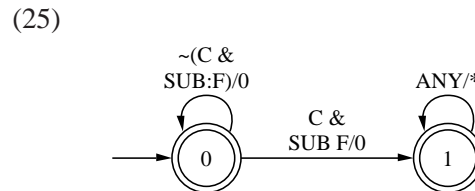


Complexity parameters (C: simple and complex) are also used as predicates in FSMs. SUB:F accepts every VI that contains a FS subsumed by

the FS F, "&" indicates logical conjunction and "~" negation. The transition from state ② to itself hence matches only VIs which are not of complexity C or do not subsume F. (23) accepts two types of strings 1) those which do not contain an appropriate F in any but the first VI. These are matched by direct traversal from the start state to state ②, and possibly following traversions from ② to itself. 2) strings which contain at least one appropriate F in a VI after the first one. The transition from ① to ② matches the last such VI. Hence all transitions from the start state to ① and on ① induce a constraint violation. Next shows the instantiation for $\text{ALIGN}([3], \text{left}, \text{simple})$:



Alignment constraints for the right edge are even simpler than the leftwards variant, and have the general form in (25):



Strings without appropriate alignment target are completely consumed on the start state, the transition from the start state to ① matches the first alignment target (if any). All following VIs (matched by transitions on ①) separate the first target from the right edge and hence induce constraint violations. Both types of alignment constraints are evaluated according to the algorithms in Ellison (1994).

4.3 COHERENCE

As GEN, COHERENCE is not implemented by a static FSM but by automata created dynamically for a given input. For reasons of space I show here only the algorithm for simple VIs. First, all indices of FSs from the input I which subsume the constraint FS are collected in the list L :

(26) **COLLECT_INDICES**(I, F)

```

 $L \leftarrow [ ]$ ;   for all  $FS$  in  $I$ :
  if  $F \sqsubseteq FS$ : insert  $index(FS)$  in  $L$ 

```

Now a transducer is created where each index from L corresponds to a state, and simple VIs with this index lead always to a transition to the corresponding state.

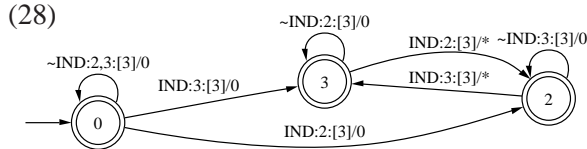
(27) **COH_AUT**(L, F)

```

Create a start state 0
Create transition:  $0 \xrightarrow{\sim\text{IND}:L:F/0} 0$ 
for all  $i$  in  $L$ 
  Create final state  $i$ 
  Create transition:
     $0 \xrightarrow{\text{IND}:i:F/0} i$ 
  Create transition:
     $i \xrightarrow{\sim\text{IND}:i(L \setminus i):F/0} i$ 
  for all  $j$  in  $L \neq i$ :
    Create transition:  $i \xrightarrow{\text{IND}:j:F/*} j$ 

```

IND $i_1 \dots i_n:F$ is a predicate which matches all simple VIs with FSs subsumed by F and bearing an index $i_1 \leq k \leq i_n$. " $L \setminus i$ " denotes the list L with all instances of i removed. (28) shows the resulting transducer for the constraint COHERENCE([3], simple) and the input in (6):



Again this transducer is evaluated according to Ellison (1994).

4.4 Relativized PARSE

Relativized PARSE constraints such as $\text{PARSE } F^{F^T/F^C}$ are implemented by the algorithm in (29), where C is the current candidate set and $I = i_1 \dots i_n$ the input. The first four lines collect indices corresponding to F^T and F^C in two lists. If both lists have length 1 (line 5), for the input FS which matches the target feature F^T and F the feature to be realized, an automaton is generated which requires that F is realized in the output corresponding to (i.e., with the index) of

the respective input FS.

(29) Evaluation Algorithm for $\text{PARSE } F^{F^T/F^C}$

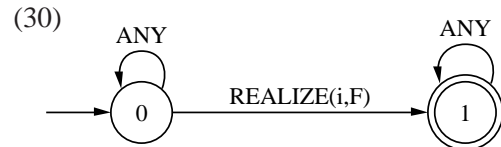
```

 $L_T \leftarrow [ ]$ ;  $L_C \leftarrow [ ]$ 
for  $j \leftarrow 1$  to  $n$ 
  if  $F, F^T \sqsubseteq i_j$ : insert  $j$  in  $L^T$ 
  if  $F^C \sqsubseteq i_j$ : insert  $j$  in  $L^C$ 
if  $length(L^C) = length(L^T) = 1$ 
   $C' = \text{PARSE\_AUT}(first(L^T), F) \cap C$ 
  if  $C' \neq \emptyset$ 
     $C \leftarrow C'$ 

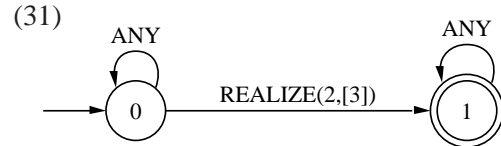
```

The three last lines are inspired by **priority union** as developed in Karttunen (1998)³: The result of intersection is only used as the new candidate set, if this does not lead to an empty candidate set.

$\text{PARSE_AUT}(j, F)$ is defined as the following automaton, which accepts every string that realizes the FS F by at least one FS with index j :



For $\text{PARSE } [3]^{[+an]/[-an]}$ and the input in (18) we get:



4.5 PARSE F

PARSE F is the most problematic constraint type discussed here since it does not count local constraint violations in the output, but computes how many input features are realized in the output. However PARSE F has a convenient property which I will call *extensional conservativity* that can be exploited for an implementation.

(32) A constraint is *extensionally conservative* if for any two candidates such that $C = C_A C_E = C'_A V C'_E$ and $C' = C_A V C_E$, C and C' induce the same number of

³In contrast to priority union the algorithm here is based on intersection, of automata, not on composition of transducers.

constraint violations (for C_A, C_E, C'_A, C'_E strings and V a VI).

In other words, for an extensionally conservative constraint candidates differing from a candidate C only by "repeating" at some place one VI already present in C are just as harmonic as C . PARSE F is extensionally conservative since by adding a VI that is already present no additional features are realized which are not already realized by the base form. Expressing features twice does neither improve nor downgrade feature realization. By transitivity all *extensions* of a form (i.e., all forms which can be built from an other form by adding an arbitrary number of VIs already present in the form) are equally optimal as the form itself.

Now conceive of the current candidate set as a regular expression (RE) RE_1 using only concatenation, disjunction the Kleene star "*" and "?" for optionality. If we replace all occurrences of "*" in RE_1 by "?" we get an expression $RE_2 = Truncate(RE_1)$ which denotes a subset of the strings denoted by RE_1 such that for each string S_1 in $(RE_1 - RE_2)$ there is at least one string S_2 in RE_2 such that S_1 is an extension of S_2 . From this and the fact that PARSE F is extensionally conservative it follows that each string in RE_1 which is optimal for PARSE F is the extension of a string in RE_2 which is also optimal for this constraint. Conversely, if a string in R_2 is optimal for PARSE F, then all its extensions in RE_1 are also optimal. Thus to find the (potentially infinite) set of optimal candidates in RE_1 , we only need to determine the (finite) set of optimal candidates in RE_2 $Optimal(RE_2)$, create the set of all its extensions $Ext(Optimal(RE_2))$ and intersect it with R_1 . Here is a summary of the procedure $Evaluate(PARSE F, R)$. R stands for an arbitrary RE and FR for a RE denoting a finite set.

1. **Truncate(R):** Return a copy of R with all occurrences of "*" replaced by "?".
2. **Optimal(FR):** Return a RE denoting all candidates from FR which are optimal for PARSE F.
3. **Ext(FR):** Return a RE which contains the set of all extensions of strings in $FR =$

$(C_1|C_2|\dots|C_n)$. Return R' which is formed by replacing each VI V in each candidate $C_i, 1 \leq i \leq n$ by ". * V .*".

4. **Evaluate(PARSE F, R):** Return $R \cap Ext(Optimal(Truncate(R)))$

References

- Bloomfield, L. (1962). *The Menomini Language*. New Haven: Yale University Press.
- Eisner, J. (1997). Efficient generation in primitive Optimality Theory. In *ACL'97*.
- Ellison, T. M. (1994). Phonological derivation in Optimality Theory. In *COLING '94*, pages 1007–1013.
- Halle, M. and Marantz, A. (1993). Distributed Morphology and the pieces of inflection. In Hale, K. and Keyser, S. J., editors, *The View from Building 20*, pages 111–176. Cambridge MA: MIT Press.
- Karttunen, L. (1998). The proper treatment of optimality in computational phonology. In *Proceedings of FSMNLP'98*.
- McCarthy, J. and Prince, A. (1993). Generalized alignment. *Yearbook of Morphology*, pages 79–153.
- McCarthy, J. and Prince, A. (1995). Faithfulness and reduplicative identity. *University of Massachusetts Occasional Papers in Linguistics*, pages 249–384.
- Prince, A. and Smolensky, P. (1993). Optimality theory: Constraint interaction in generative grammar. Technical reports of the Rutgers University Center of Cognitive Science.
- Trommer, J. (1999). Generation and parsing in OT-based morphology. In *Proceedings of 5th Formal Grammar Conference, Utrecht, August, 1999*.
- Trommer, J. (2002). Hierarchy-based competition. Ms., available under <http://www.ling.uni-osnabrueck.de/trommer/hbc.pdf>.
- Trommer, J. (2003). The interaction of morphology and syntax in affix order. In Booij, G. and van der Marle, J., editors, *Yearbook of Morphology 2002*, pages 283–324. Dordrecht: Kluwer.