Some Phonological Phenomena in Korean: an Optimality Theoretic Account*

Seok-keun Kang
(Wonkwang University)

Kang, Seok-keun (1995) **Some Phonological Phenomena in Korean: an Optimality Theoretic Account.** *Linguistics* vol. 3. The purpose of this paper is to reconsider obstruent neutralization and consonant cluster simplification in Korean within the framework of Optimality Theory. Considering the interaction of the Basic Syllable Structure Constraints proposed by Prince and Smolensky (1993), first, I claim that in Korean, ONS and -COD are dominated by FILL and PARSE. Second, I consider some theoretical devices for handling obstruent neutralization and consonant cluster simplification in an Optimality Theory framework, and assert that these phonological phenomena can be accounted for in a unified way in terms of a ranking of constraints.

1. Introduction

In this paper, I will consider the pattern of obstruent neutralization and consonant cluster simplification in Standard Korean within the framework of Optimality Theory (Prince & Smolensky 1993, and McCarthy & Prince 1993). So far, there has been in the literature much discussion - largely rule-driven analyses (to name a few, Hong 1982, Whitman 1985, Kim 1986) - on these phonological phenomena, none of which, however, provides a satisfactory explanation. By employing the framework of Optimality Theory, in what follows, I will show that a constraint-based approach can give a better account of these phenomena.

The paper is organized as follows. Section 2.1 provides a brief introduction to the principles and assumptions of Optimality Theory. In section 2.2, I explicate the interaction of the Basic Syllable Structure Constraints proposed by Prince and Smolensky (1993) in Korean phonology. I propose in the next two sections more adequate treatments of obstruent neutralization and consonant cluster simplification in Optimality Theory, arguing for a special constraint ranking which produces correct outputs. A brief conclusion is given in section 3.

2. Obstruent neutralization and consonant cluster simplification

2.1. Optimality Theory

Optimality Theory is a purely constraint-based approach to phonological well-formedness. In this theory, Universal Grammar provides a set of highly general well-formedness constraints, which are in principle violable, and an individual grammar consists of a ranking of these constraints; languages differ primarily in how they rank these often conflicting constraints in strict dominance hierarchies that determine the circumstances under which constraints are violated.

Optimality Theory shifts the burden of explanation from derivation to constraint systems, and so there are no phonological rules, repair strategies and other derivational notions. Instead, the grammar is schematically represented as follows:

(1) a. Gen (In_k) $\rightarrow$ {Out_1, Out_2, ...}
    b. H-eval (Out_i, $1 \leq i \leq \infty$) $\rightarrow$ Out_{real}

The function Gen emits an infinitely large set of possible candidate analyses consistent with a given input. Then, H-eval determines the relative harmony of the candidates in terms of a hierarchy of constraints whose relative priority is determined by the language. An optimal output is at the top of the harmonic order on the candidate set; i.e., it is the one that best satisfies the constraint hierarchy. To illustrate, consider the tableau in (2), which lays out the evaluation procedure. Assume a set of ranked constraints C_1, C_2, C_3 and C_4, and a candidate set {O_1, O_2, O_3} generated from a given input. In this tableau, and henceforth, constraints are ordered left to right in order of priority. Violations are marked by "*", and fatal violations are also signalled by "!". The pointing hand "☞" indicates optimal candidates. In addition, cells that do not participate in the decision are shaded.

(2)

Candidates	C_1	C_2	C_3	C_4
O_1	*!	▨	▨	▨
☞ O_2		*		▨ *
O_3		*	*!	▨

Evaluation of the candidates proceeds recursively by constraint. Candidate 1 violates the top-ranked constraint C_1, so it is excluded from consideration immediately, because of the other alternatives which pass it. That is, if a candidate fails a constraint that at least one other candidate meets, the violation is critical. Candidates 2 and 3 are not distinguished by constraint C_2, and so constraint C_3, the next constraint down the hierarchy, becomes relevant. Candidate 2 obeys but candidate 3 violates constraint C_3. Hence, candidate 2 emerges as optimal. Constraint C_4 has no bearing on the outcome. Note that in (2), candidate 2 is selected as optimal even though it contains one more violation than candidate 1; this implies that what is crucial in deciding optimal outputs is not the number of but the ranking of the constraints that candidates violate. I refer the reader to Prince and Smolensky (1993) for the comprehensive view and formalization.

Given this much theoretical background, in what follows, I will turn to an optimality theoretic account of Korean consonants.

2.2. Korean syllable structure

The Basic Syllable Structure Constraints divide notionally into two groups: those that describe the universally unmarked characteristics of the structures involved (e.g., ONS in (3) and -COD in (4)) and those that constrain the relation between structure and input (e.g., PARSE in (5) and FILL in (6)) (Prince & Smolensky 1993).

(3) ONS: Syllables must have onsets.

(4) -COD: Syllables must not have a coda.

(5) PARSE: Underlying segments must be parsed intosyllable structure.

(6) FILL: Syllable positions must be filled with underlying segments.

The constraint ONS urges that a syllable node have an Ons node; the presence of the Ons node satisfies ONS whether or not that node is filled. The constraint -COD requires that a syllable node have no Cod child; the presence of a Cod node violates -COD whether empty or filled. PARSE and FILL declare that perfectly well-formed syllable structures are those which are faithful to the input strings, in the sense of a one-to-one correspondence between syllable positions and segments.

According to Prince and Smolensky (1993), the constraints (3-6) may be relatively ranked in any dominance order in a particular language. The optimal analysis for a given language is determined by whichever of the constraints given above is lowest in the constraint hierarchy of that language. In this section, I will explicate the interaction of the constraints in (3, 4) with those in (5, 6) in Korean phonology. The Korean syllable structure is maximally CGVC; onsets and codas are optional. As Prince and Smolensky (1993) assert, in this kind of languages, ONS and -COD are dominated by PARSE and FILL. Thus , in Korean, .V. is the optimal parse of /V/, as exemplified in (7). Here and throughout, columns separated by a dotted line (such as FILL and PARSE in (7)) do not conflict and so are not crucially ordered. " □ " stands for an empty node, < > shows an unparsed segment, and periods mark the edges of syllables.

(7) Input /o/ 'five'

Candidates	FILL	PARSE	ONS
a. ☞ .o.			*
b. <o>		*!	
c. .□o.	*!		

Each of these candidates involves a violation of one of the Basic Syllable Structure Constraints. Candidate .o. fails ONS, while candidates <o> and .□o. violate PARSE and FILL, respectively. Because .o. is the actual output form, the grammar requires that constraints FILL and PARSE dominate ONS, in the sense that the lowest constraint incurs the least violation. That is, Korean does not absolutely require onsets. In (7), the relative ranking of FILL and PARSE has no bearing on the outcome. However, the violations of these constraints are fatal, because the alternative candidate .o. meets both constraints.

Now, let's consider the input /CVC/, of which the final C induces the conflict between FILL, PARSE and -COD. As in the situation of the onset, the optimal analysis of /CVC/ in a given language is determined by which of the three constraints ranks lowest in the dominance hierarchy. In Korean, .CVC. wins, so the violation of -COD is compelled, as shown in the following tableau:

(8) Input /kak/ 'angle'

Candidates	FILL	PARSE	-COD
a. ☞ .kak.			▨ *
b. .ka<k>		*!	▨
c. .ka.k □ .	*!	▨	▨

Here candidate .kak. meets FILL and PARSE, but fails -COD. The other alternatives .ka<k>. and .ka.k □. pass -COD, while they violate PARSE and FILL, respectively. In (8), .kak. is the optimal candidate, so -COD is the lowest of the three constraints, which means that in Korean, a Cod node is not forbidden but optional.

To summarize this section, I propose that Korean has the following set of ranked constraints, where A » B means A is more highly ranked than B.

(9) Interim Constraint Hierarchy:
FILL, PARSE » ONS, -COD

Henceforth, I will not comment on the violation marks *ONS and *-COD, since ONS and -COD violations do not play a decisive role in the competitions of interest.

2.3. Obstruent neutralization

This section deals with obstruent neutralization in the framework of Optimality Theory.[1] To begin with, the set of possible codas in Korean is restricted to [p, t, k, m, n, ŋ, l]. Place, [nasal] and [lateral] features are distinctive in the coda, but others are not. It is also well-known that obstruents with laryngeal or other features (e.g., manner features) may appear syllable-finally in the underlying representation, but they surface neutralized, as shown in (10).

(10) Obstruent Neutralization[2]

a. $\{p, p', p^h\} \rightarrow [p] / \_\_\_]\sigma$

b. $\{t, t', t^h, s, s', c, c', c^h\} \rightarrow [t] / \_\_\_]\sigma$

c. $\{k, k', k^h\} \rightarrow [k] / \_\_\_]\sigma$

As seen above, codas in Korean are subject to strong limitations of the familiar kind. The generalization is that codas may be occupied only by plain voiceless stops, nasals and /l/. I formulate the relevant constraint as follows:

(11) CODA-COND: Coda licenses only Place, [nasal], [lateral].

I further assume that Gen (the function that generates output candidates from a given input) has the power to delete features (i.e., underparsing of features), but that deletion of features incurs a cost, namely violation of the following featural constraint:

(12) PARSE-FEAT: Underlying features must be parsed.

The constraint PARSE-FEAT is a constraint against nonparsing of underlying features; delinked features count as PARSE-FEAT violations. The tableau in (13), for example, shows how CODA-COND and PARSE-FEAT conspire with other constraints to produce the optimal output. Since the four candidates violate one constraint each, any comparison among them will involve weighing the importance of different violations. In the case at hand, .ap. is the optimal output, so PARSE-FEAT is the lowest-ranked of the constraints. The relative ranking of CODA-COND, FILL and PARSE has no effect on the outcome. Note that in (13), the violation of CODA-COND is avoided by failure to meet the constraint PARSE-FEAT.

(13) Input /ap^h/ 'front'

Candidates	CODA-COND	FILL	PARSE	PARSE-FEAT
a. .ap^h.	*!			
b. ☞ .ap. $\overset{\bullet}{}$ ǂ [+s.g.]				*
c. .a<p^h>.			*!	
d. .a.p^h □ .		*!		

As discussed above, PARSE-FEAT must be dominated by CODA-COND, FILL and PARSE; otherwise an incorrect output should be selected as optimal. If we were to reverse the domination ranking of PARSE and PARSE-FEAT, for example, an incorrect output would be produced, as exemplified in the following tableau:

(14) Input /ap^h/ 'front' PARSE-FEAT » PARSE

Candidates	CODA-COND	FILL	PARSE-FEAT	PARSE
a. .ap^h.	*!			
b. .ap. + [+s.g.]			*!	
c. .a<p^h>.				*
d. .a.p^h □ .		*!		

In (14), .a<p^h>. would be incorrectly claimed to be optimal. That the analysis in (14) is not correct implies that the violation of PARSE by .a<p^h>. is worse than the violation of PARSE-FEAT by .ap.; i.e., we must have PARSE dominating PARSE-FEAT. The required ranking is recorded here:

(15) Interim Constraint Hierarchy:
CODA-COND, FILL, PARSE » PARSE-FEAT

This ranking has been justified by the arguments given immediately above. As a check on the correctness of the analysis, I provide tableau (16). Since CODA-COND refers solely to parsed features, CODA-COND can be satisfied trivially leaving a feature unparsed. Thus, in the winning candidate in (16), [+continuant] is left unparsed, and since PARSE-FEAT is the lowest-ranked of the relevant constraints, this is the optimal output. The tableau in (16) shows that when CODA-COND has priority than PARSE-FEAT, PARSE-FEAT may be violated by the optimal candidate to gain success on CODA-COND.

(16) Input /nas/ 'sickle'

Candidates	CODA-COND	FILL	PARSE	PARSE-FEAT
a. .nas.	*!			
b. ☞ .nat. + [+cont]				*
c. .na<s>.			*!	
d. .na.s□.		*!		

2.4. Consonant cluster simplification

Let's turn now to the cases where consonant clusters occur in coda position. In Korean, when two consonants occur in coda underlyingly, one of them is deleted (i.e., unparsed), as exemplified in (17); the segments in the parentheses are deleted in coda position.[3]

(17)

a. p(s):	/kaps/ → [kap]	'price'
k(s):	/nəks/ → [nək]	'soul'
n(c):	/anc-ta/ → [ant'a]	'to sit down'
b. (l)k:	/ilk-ta/ → [ikt'a]	'to read'
(l)p:	/palp-ta/ → [papt'a]	'to tread'
(l)m:	/salm-ta/ → [samt'a]	'to boil'
c. l(t^h):	/halt^h-ta/ → [halt'a]	'to lick'

Here, the relevant constraint is *COMPLEX in (18), which says that syllable positions are limited to single segments.

(18) *COMPLEX: No more than one C or V may associate to any syllable position node. (Prince & Smolensky 1993)

The constraint *COMPLEX plays a decisive role in producing the optimal outputs in (17). That is, input sequences CVCC are resolved as either CVC<C> or CV<C>C, incurring a *PARSE violation, as seen in the disappearance of /s/ in /kaps/ 'price'.

First, let's consider the examples in (17a). In the preceding section, I have assumed that FILL and PARSE are unranked. In fact, however, the relative ranking between these constraints is crucial to correctly determine optimal candidates based on inputs such as /kaps/, as shown in (19). In Korean, *COMPLEX, CODA-COND and FILL are unviolated, and hence undominated in the constraint hierarchy. In (19), candidates (a, c, e) violating these three constraints are excluded from consideration immediately, since non-violating candidates exist. Here is the sense in which there are no trade-offs: no amount of respect paid to subordinate constraints can rescue these candidates in the eyes of the theory (Prince & Smolensky 1993, Ito et al. 1993). The two survivors, candidates (19b) and (19d), tie in failing constraint PARSE; hence, neither violation is critical. The decision between them must be passed on to the subordinate constraint. (19b) passes but (19d) fails constraint PARSE-FEAT, and so candidate (19b) emerges as optimal. In the winning candidate, since /s/ is not parsed (i.e., unassociated to a syllable position), it is not phonetically realized.

(19) put /kaps/ 'price'

Candidates	*COMPLEX	CODA-COND	FILL	PARSE	PARSE-FEAT
a. .kaps.	*	*!			
b. ☞.kap<s>.				*	
c. .ka<p>s.		*!		*	
d .ka<p>t. ↑ [+cont]				*	*!
e. .kap.s □ .					

The tableau above illustrates the interaction of *COMPLEX, CODA-COND, FILL, PARSE and PARSE-FEAT in selecting optimal candidates. The required ranking of the constraints is as follows:

(20) Interim Constraint Hierarchy:
*COMPLEX, CODA-COND, FILL » PARSE » PARSE-FEAT

Note that in (19), FILL should be ranked over PARSE. If FILL is dominated by PARSE, then an incorrect outcome will be produced, as illustrated in the following tableau; here, .kap.s □ . will be wrongly selected as optimal:

(21) Input /kaps/ 'price' PARSE » FILL

Candidates	*COMPLEX	CODA-COND	PARSE	FILL	PARSE-FEAT
a. .kaps.	*	*!			
b. .kap<s>.			*!		
c. .ka<p>s.		*!	*		
d .ka<p>t. ǂ [+cont]			*!		*
e. .kap.s □ .				*	

So far, I have argued that the fact of consonant loss in (17a) can be accommodated under the constraint hierarchy in (20).

Now let's consider the examples in (17b). For the case at hand, a further constraint *M/l in (22), comes into play, for in that case parsing /l/ violates *COMPLEX even though /l/ is also a possible coda.

(22) *M/l: /l/ must not be parsed as a syllable margin.

The following tableau, for example, shows how the constraint *M/l conspires with the constraints discussed above to produce the correct output .pap.ta. from its input /palpta/. Candidates (23a) and (23d) fail top-ranked constraints *COMPLEX and FILL, respectively, and so they are eliminated from consideration. The other alternatives (23b) and (23c) obey the top-ranked constraints. They also contain equal numbers of PARSE violations. The tie between these otherwise equally harmonic candidates is broken by *M/l, which rules out (23c). That is, the actual output is (23b). Here, PARSE-FEAT is irrelevant for the decision of the optimal output; it cannot break the tie between the two main contenders.

(23) Input /palpta/ 'to tread'

Candidates	*COM-PLEX	CODA-COND	FILL	PARSE	PARSE-FEAT	*M/l
a. .palp.ta.	*!					*
b. ☞.pa<l>p.ta.				*		
c. .pal<p>.ta.				*		*!
d .pal.p □ .ta.			*!			*

Note that .pa<l>p.ta. in (23b) would still be selected as optimal under the ranking *M/l » PARSE or *M/l » PARSE-FEAT. But evidence that neither ranking is correct is provided by the analysis of the examples in (17c). /halt^{h}ta/, for example, is analyzed as in (24).

(24) Input /halt^{h}ta/ 'to lick'

Candidates	*COM-PLEX	CODA-COND	FILL	PARSE	PARSE-FEAT	*M/1
a. .halt^h.ta.	*	*!				*
b. .ha<l>t^h.ta.		*!		*		
c. ☞.hal<t^h>.ta.				*		*
d .ha<l>t.ta. + [+s.g.]				*	*!	
e. .hal.t^h □ .ta.			*!			*

In (24), only the candidates which fare best by the dominant constraints *COMPLEX, CODA-COND and FILL survive for further consideration. Candidates (a, b) and (e) fail these constraints, which is fatal, because of the competing candidates which satisfy them. The remaining two parses (c) and (d) are not distinguished by the top-ranked constraints *COMPLEX, CODA-COND and FILL. They also tie in violating high- ranking PARSE. Therefore, the next constraints down the hierarchy become relevant. Candidate (c) fares worse on *M/l, but it crucially obeys dominant PARSE-

FEAT, which candidate (d) violates; hence, (c) is the actual output.

Tableau (24) above shows that we must rank the constraint *M/l at the bottom of the hierarchy below PARSE- FEAT, as in (25).

(25) Final Constraint Hierarchy:
*COMPLEX, CODA-COND, FILL » PARSE » PARSE-FEAT » *M/l

If we rank *M/l over either PARSE-FEAT or PARSE, then an incorrect output emerges as optimal. Assume, for example, that *M/l outranks PARSE-FEAT. Then, as shown in the following tableau, *.ha<l>t.ta. should be optimal. It is clear from (26) that we should have PARSE-FEAT (and PARSE) dominating *M/l.[4]

(26) Input /halt^hta/ 'to lick' *M/l » PARSE-FEAT

Candidates	*COM-PLEX	CODA-CON	FILL	PARSE	*M/1	PARSE-FEAT
a. .halt^h.ta.	*	*!			*	
b. .ha<l>t^h.ta.		*!		*		
c. .hal<t^h>.ta.				*	*!	
d .ha<l>t.ta. † [+s.g.]				*		*
e. .hal.t^h □.ta.			*!		*	

In this section, I have shown that consonant cluster simplification can be given a natural account in the framework of Optimality Theory; i.e., these phonological processes can be accounted for in a unified way in terms of the ranking of constraints in (25).[5]

3. Conclusion

To sum up, first, I have considered Korean syllable structure in terms of Optimality Theory, claiming that in Korean, FILL and PARSE are more highly ranked than ONS and -COD; that is, onsets and codas are not required/forbidden but optional. Second, I have also considered some theoretical devices for handling obstruent neutralization and consonant cluster simplification, and have claimed that the actual Korean parse is indeed optimal as determined by the constraint hierarchy (25).

The optimality theoretic account of obstruent neutralization and consonant cluster simplification offered in this paper is preferred over previous accounts in that it abolishes phonological rules which sometimes must apply in an extrinsic order in previous accounts, and it also obviates the need to build the ill-formed intermediate structures which are sometimes inevitable in earlier approaches. What I have asserted is that obstruent neutralization and consonant cluster simplification in Korean is directly related to a hierarchy of some ranked and violable constraints.

Notes

*This work was supported by Wonkwang University Research Fund 1995. An earlier version of the paper was presented at International Conference on Linguistics, Seoul, Korea, July 3-7, 1995. I would like to thank Young-Seok Kim, Kee-ho Kim, Yongsoon Kang, and Ongmi Kang for their valuable comments and suggestions. Any errors are, of course, my own responsibility.

1. The consonants of Korean are given below.

<u>Lexical Consonants of Korean</u>

lax	p	t	s	c	k	
tense	p'	t'	s'	c'	k'	
aspirated	p^h	t^h		c^h	k^h	h
nasal	m	n			ŋ	
lateral		l				

2. In Korean, consonant /h/ has been subject to much controversy. Past works on the behavior of /h/ in coda position can be divided into two groups: (i) those claiming that /h/ is unreleased to [t] in syllable-final position (e.g., Kim-Renaud 1974); (ii) those asserting that its realization is determined by the adjacent segment (e.g., Kim 1989). Discussion of /h/ is beyond the scope of this paper, so I will leave it for further research.

3. In Korean, an obstruent becomes tensed when preceded by another obstruent, as exemplified in (i).

(i) a. /kaksi/ → [kaks'i] "bride"
b. /ip-ta/ → [ipt'a] "to wear"
c. /pap#pəli/ → [papp'əli] "rice#earning (work)"

Since the tensing rule is a post-lexical rule, it applies regardless of the presence/absence of, and types of boundaries; i.e., the rule applies in (ia) which is not a derived environment, in (ib) which involves a suffixation, and in (c) which is a compound noun. In this paper, I will not discuss tensification. For a detailed analysis, see Ahn 1985, Kim 1986, Kim 1989 among others.

4. The only true lexical counterexample to this generalization is /ilp^h-ta/ 'to recite', which is phonetically realized as [ipt'a] rather than *[ilt'a]. In order to produce the correct output in the case at hand, we should rank *M/l more highly than PARSE-FEAT. Note, however, that /ilp^h-/ is the only lexical stem which ends in /lp^h/ cluster.

5. In Southeastern dialect of Korean, a different simplification phenomenon occurs. Observe the following forms in (i), especially the variant consonant cluster simplification in (ib) and (ic).

(i)	a.	p(s):	/kaps-to/ →	[kapt'o]	'the price also'
		k(s):	/nəks-to/ →	[nəkt'o]	'the soul also'
		n(c):	/anc-ta/ →	[ant'a]	'to sit down'
		l(t^h):	/halt^h-ta/ →	[halt'a]	'to lick'
		(l)m:	/salm-ta/ →	[samt'a]	'to boil'
	b.	l(p^h):	/ilp^h-ta/ →	[ilt'a]	'to recite'
	c.	l(k):	/ilk-ta/ →	[ilt'a]	'to read'
		l(p):	/palp-ta/ →	[palt'a]	'to tread'

The forms in (ia) are the same as in Standard Korean. But difference between the Southeastern dialect of Korean and Standard Korean appears in (ib) and (ic), in which the forms show that unlike in Standard Korean, /l/ in 'lp^h, lk and lp' surfaces, and the second consonant is deleted. The forms in (ia) and (ib) can be accommodated under the constraint hierarchy (25). In order to account for (ic), however, we need to add two more constraints, *M/k and *M/p, and rank them over *M/l.

References

Ahn, S. C. (1985). *The Interplay of Phonology and Morphology in Korean*. Doctoral Dissertation. University of Illinois at Urbana-Champaign.

Hong, C. S. (1982). "A study of syllable-final double consonants in Korean (in Korean)." *Emun-Nonjip* [Papers in Language and Literature] 23, 475-493.

Itô. J., A. Mester and J. Padgett. (1993). "Licensing and redundancy: underspecification in Optimality Theory." ms., UC Santa Cruz.

Kim, J. M. (1986). *Phonology and Syntax of Korean Morphology*. Doctoral Dissertation. University of Southern California.

Kim, S. H. (1989). "The behavior of /h/ in Korean." *Harvard Studies on Korean Linguistics III*, 117-126.

Kim, Y. S. (1984). "On the treatment of consonant cluster reduction." *Language Research* 20, 345-366.

Kim-Renaud, Y. K. (1974). *Korean Consonantal Phonology.* Doctoral Dissertation. University of Hawaii.

McCarthy, J. J. and A. Prince. (1993). "Generalized Alignment." ms., University of Massachusetts, Amherst, and Rutgers University, New Brunswick, N.J.

Prince, A. and P. Smolensky. (1993). *Optimality Theory: Constraint Interaction in Generative Grammar.* ms., Rutgers University, New Brunswick, and University of Colorado, Boulder. [forthcoming MIT Press.]

Whitman, J. (1985). "Korean clusters." *Harvard Studies on Korean Linguistics II,* 280-290.

Seok-keun Kang
Department of English Language and Literature
Wonkwang University
344-2, Shinyong-dong
Iksan, Chonbuk 570-749