Final Obstruent Gemination in Japanese Loanwords

Derek Andersen

March 12, 2020

1 Introduction

Japanese loanwords provide a rich source of data for the analysis of the gemination of certain consonantal segments in Japanese, and in particular, the types of segments for which Japanese phonology allows gemination. In this paper I will provide a computational analysis of the process of gemination in some Japanese loanwords, drawing on formulas of First Order logic. My analysis will focus particularly on the process of word-final obstruent gemination, e.g. /p/, /t/, /k/, /b/, /d/, and /g/. As we will see, in addition to the type of segment in question, the length of the preceding vowel also plays a role in whether gemination is licensed. In section 2, I introduce the process of word-final obstruent gemination and provide some examples, and in section 3, I present a logical analysis for the gemination transformation using First Order logic with the successor relation.

2 The gemination process

The native phonemic inventory of Japanese consists of singleton consonants (like /p/, /t/, /k/) as well as geminate consonants (like /pp/, /tt/, /kk/). As shown by the minimal pair data in 1, they are contrastive, and thus separate phonemes. [Kub09]

(1) a /kita/ 'came' vs. /kitta/ 'cut (past)'

b /saki/ 'point' vs. /sakki/ 'a short time ago'

Gemination in Japanese loanwords behaves similarly to its native counterpart process, with an extension to allow for voiced obstruent gemination (which is not attested natively). Specifically, words like 'red' (assuming an input form of /red/ in Japanese) when borrowed into Japanese will have their final consonant geminated (in this case, to [reddo]) in the output. For the purposes of my analysis, I will disregard the final vowel epenthesis in all cases of loanword phonemicization, as it does not speak to the gemination process. Examples of word-final obstruent gemination cases are shown in 2. [Kaw15]

(2) a 'ca**t**' → [ka**tt**o]

b 'pa**ck**' → [pa**kk**u]

c 'bi**g**' → [bi**gg**u]

There is also a stipulation which states that for word-final obstruent gemination to occur, the preceding vowel must be short in length. In other words, word-final obstruent gemination is not licensed when the preceding vowel is long, as in the examples in 3. [Kub09]

(3) a 'mea**t**' → [mii**t**o] *[mii**tt**o]

b 'pea**k**' → [pii**k**u] *[pii**kk**u]

c 'par**k**' → [paa**k**u] *[paa**kk**u]

3 Logical analysis

3.1 Defining a model for 'pack'

For the purposes of this logical analysis of the transformations at work, I will assume the input to be the surface phonetic form of the word in the original language (in this case, English). Importantly, I will be ignoring the vowel change between [æ] → [a] in 'pack', shown below in 4. This phonemicized form will be used as our input for the word-final obstruent gemination transformation. Again, the final vowel epenthesis is not being considered for this analysis and should be disregarded.

(4)

	Source		**Phonemicized**		**Output**
'pack'	/pæk/	→	/pak/	→	[pakk(u)]

Thus, considering the word 'pack', the input for the transformation for this word will be /pak/. Our goal will be to define a logical model for a transformation that will account for the output [pakk] from the input /pak/. For this purpose I will be utilizing a model of First Order logic with the successor relation, and its signature is in 5. Included in this signature are the unary relations p, a, and k because they are necessary to express the transformation from input to output, as we will see. Omitted for readability are the remaining segments of Japanese, including but not limited to the obstruents that would undergo gemination in the same context (/t/, /b/, /d/, /g/) as well as other vowels.

(5) $M^{\triangleleft} := \langle D; \triangleleft, p, a, k \rangle$

The transformation that I will be using to account for the gemination process is a copying transformation which relies on the use of a copyset, C. Through the use of a copyset, the input (copy 1) can be "copied" into the output (copy 2), and the formulas I define will license the transformation in question. In section 3.2, the use of copies will make clear the intuition behind the choice of using the successor relation in our model.

The first formulas that I will define as part of my logical model will be the unary relations for p, a, and k. These are listed in 6. To denote the formulas which refer to the relations in copy 2 vs. copy 1, I use a superscript ^{2}. These copies will be discussed further in section 3.2.

(6)
a $\varphi_p^1(x) := p(x)$

b $\varphi_p^2(x) := False$

c $\varphi_a^1(x) := a(x)$

d $\varphi_a^2(x) := False$

e $\varphi_k^1(x) := k(x)$

f $\varphi_k^2(x) := \texttt{obstruent}(x) \wedge \texttt{last}(x) \wedge \texttt{after\_short}(x)$

Using 6a as an example, these formulas state something along the lines of the following: "If x is a p in the input, it will be a p in the output (in this case, of copy 1)." This necessarily means that in the cases of 6b and 6d, the formulas state that input ps and as will not carry over to the output (of copy 2). Furthermore, according to 6f, it is the case that input ks will appear in the output (of copy 2) iff the user-defined predicates `obstruent`, `last`, and `after_short` defined in 7, 8, and 9 are satisfied by x.

(7) $\texttt{obstruent}(x) := p(x) \vee t(x) \vee k(x) \vee b(x) \vee d(x) \vee g(x)$

(8) $\texttt{last}(x) := \neg\exists y(x \lhd y)$

(9) $\texttt{after\_short}(x) := \exists y(\texttt{short\_vowel}(y) \wedge y \lhd x)$

(10) $\texttt{short\_vowel}(x) := a(x) \vee i(x) \vee u(x) \vee e(x) \vee o(x)$

Predicate 7 states that if x is any of the symbols p, t, k, b, d, g, then x is an obstruent, while predicate 8 states that if there is no such y such that y is the successor of x, then x is the last symbol in the string. Predicate 9 draws upon predicate 10 to state that if there exists a y such that y is a short vowel and x succeeds y, then x has the property of being after a short vowel, importantly giving us the power to express which symbols will be licensed for gemination. Note here that "short" refers to vowels which are not "long", denoted by a double /aa/ vs. a single /a/, for example.

3.2 The copyset and binary relations

The mechanism which will make my analysis of the gemination process possible is the copyset, C, defined in 11.

(11) $C := \{1, 2\}$

Each element in the copyset will generate an output copy of our input string, 'pak', which is illustrated in Figure 1.

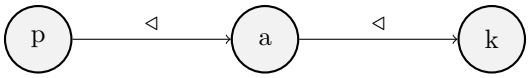

Figure 1: Input representation of 'pak' under our successor model

Before copies 1 and 2 can be introduced, I must define the formulas for the binary successor relation in our model for 'pak'. These are in 12, and they are denoted by superscript pairs of 1 and 2, which indicate the directionality of the successor relation for the arguments in the formula.

(12) a $\varphi_{\triangleleft}^{1,1}(x, y) := x \triangleleft y$

b $\varphi_{\triangleleft}^{1,2}(x, y) := \texttt{last}(x) \wedge \texttt{last}(y) \wedge x \triangleleft y$

c $\varphi_{\triangleleft}^{2,1}(x, y) := False$

d $\varphi_{\triangleleft}^{2,2}(x, y) := x \triangleleft y$

Now we can look at the representation of the output strings for copies 1 and 2, and drawing upon the formulas defined in 6 (the unary relations) and 12 (the binary relations), we can explain the transformation taking place which results in /pak/ → [pakk]. Crucially, what our binary relation formulas do is provide criteria for which successor relations will hold in the final output, which can be thought of as a "combination" of copy 1 and copy 2. The formulas in 12 are clarified below.

12a Across members of copy 1, members which are in the successor relation will remain so

12b Between members of copy 1 → copy 2, if x is the last element of a copy and y is the last element of a copy, then x is succeeded by y

12c Between members of copy 2 → copy 1, no successor relations hold

12d Across members of copy 2, members which are in the successor relation will remain so

What this entails is that in copy 2 (shown in Figure 3), ø is succeeded by ø which is succeeded by *k*, which is necessarily true if 'ø' is treated as *nothing*. It is precisely for this reason that 12b holds, which allows the successor relation from copy 1 → copy 2 to "skip" positions 1 and 2 in copy 2, and consider only position 3 which is occupied by the *k*.

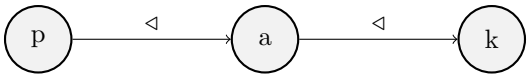

Figure 2: Output representation of copy 1 of 'pak'

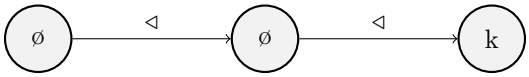

Figure 3: Output representation of copy 2 of 'pak'

The final two formulas that are needed are ones that will allow, or license, the segments which are acceptable in the output [pakk]. Since we have two copies, we will need to define two separate licensing formulas, one for each copy. These are defined in 13.

(13) a $\varphi^1_{license}(x) := True$

b $\varphi^2_{license}(x) := \texttt{obstruent}(x) \wedge \texttt{last}(x)$

The formula in 13a simply states that segments in copy 1 are always licensed. In 13b, however, the formula restricts licensed segments to those which satisfy `obstruent` and `last`. In the case of [pakk], the first [p], [a], and [k] are elements of the first copy, and thus are all licensed, and the second [k] is (1) a member of the second copy, (2) an obstruent, and (3) the last segment of its copy. For these reasons, the final [k] is licensed. Crucially, what allows for the correct order of the output segments is the binary successor relations defined in 12. The final output representation of [pakk] is illustrated in Figure 4.

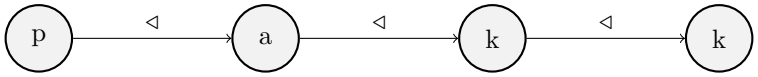

Figure 4: Final output representation of [pakk]

4 Conclusion

We have seen that in certain contexts, Japanese loanwords undergo word-final obstruent gemination. Specifically, when the preceding vowel is short, and the obstruent in question is in the final position in the input string, gemination is licensed. I have presented in this

paper a logical analysis of a transformation accounting for this gemination process, drawing upon formulas of First Order logic with the successor relation. It is apparent that in order for our model to successfully account for the process, the successor relation is necessary — it is what allows us to define binary relations between elements of the first copy to the second copy, and account for gemination in the surface form.

References

[Kub09] Haruo Kubozono. "Consonant Gemination in Japanese Loanword Phonology". In: *Current Issues in Unity and Diversity of Languages, The Linguistic Society of Korea* (2009), pp. 953–973.

[Kaw15] Shigeto Kawahara. "Geminate devoicing in Japanese loanwords: Theoretical and experimental investigations". In: *Language and Linguistics Compass* 9 (2015).