Rule Based Grapheme to Phoneme Mapping for Hindi Speech Synthesis

1 author:

Monojit Choudhury
Microsoft
99 PUBLICATIONS **926** CITATIONS

SEE PROFILE

Section: Computer Science

Rule Based Grapheme to Phoneme Mapping for Hindi Speech Synthesis*

Monojit Choudhury
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

1. Introduction

Speech is one of the most vital forms of communication in our everyday life. On the contrary, the dependence of human computer interaction on written texts and images, makes the use of computers impossible for visually and physically impaired and illiterate masses. Automatic speech generation from natural language sentences can overcome these obstacles. This work is part of a larger project aiming at developing communication tools in Indian languages for the visually handicapped and cerebral palsy affected children.

Formally, a *Text to Speech* (*TTS*) [**1**,**2**] system converts a written text to speech or a sound file. The goal of a *TTS* system is to provide intelligible and natural speech. A *TTS* system consists of a *Natural Language Processing (NLP)* module and a *Digital Signal Processing (DSP)* module. The *NLP* module converts the text, that is the *graphemes,* to a string of *phonemes* [**3,4**]. It also encodes the *intonation* and *prosodic* information in the output string. In a concatenative synthesis approach, the *DSP* module obtains the sound files from an acoustic inventory corresponding to the string of *phonemes* or *diphones*[1] and concatenates them. Finally, it modulates the sound according to the *intonation* and *prosodic* information. Since in almost all languages, we hardly pronounce what we write, some linguistic analysis is necessary for the intelligibility of the speech.

This paper describes a rule based grapheme to phoneme mapping for Hindi. Section 2 introduces the different issues involved in Hindi phonology which affect the grapheme to phoneme mapping. The next three sections describe in depth algorithms for three most important subproblems in Hindi phonology viz. *schwa deletion*, marking the *syllable boundaries* and pronunciation of the diacritical marks ‘ ’ (*anusvara*) and ◌ॅ . Section 6 concludes this paper and discusses the implementation issues.

2. Issues in Hindi Phonology

Having its root in Sanskrit, which is phonetically perfect (i.e. there is very little or almost no discrepancy between written text and pronunciation), Hindi is pronounced almost as it is written. Going by the *Optimality Theory* [**5**], this can be restated as “In Sanskrit and Hindi the *faithfulness constraints* are ranked higher than the *markedness constraints*”. Even then there are several issues in Hindi phonology that needs to be

* This work has been done under the supervision of Prof. Anupam Basu, CSE, IIT Kharagpur and was partly supported by Media Lab Asia project funding.

[1] The acoustic inventory may store phonemes, diphones, triphones, or even syllables. This partly depends on the language and partly on the technique chosen for synthesis.

sorted out before we can construct a grapheme to phoneme converter for Hindi, some of which are listed below.

- Schwa deletion
- Sound of anusvara and chandra-bindu
- Syllable boundary marking
- Context dependent pronunciation of schwa
- Special sounds for conjugates like ***j~na*** [ज्ञ] and vowel ***R^i*** [ऋ]

Each consonant in written Hindi is associated with an "inherent" *schwa*[2], which is not explicitly represented. Other vowels are overtly written diacritically or non-diacritically around the consonant. The problem is that *schwa* is sometimes pronounced and sometimes not. For example, in the word ***dha.DakaneM*** [धड़कनें, dhəɽ.kən.ẽ, noun. heart-beats], the *schwa* following ***.D*** is deleted in the pronunciation. Schwa deletion along with the word-morphology determines the syllable boundaries in a word and hence is a vital part of Hindi phonology. Just to illustrate how improper *schwa deletion* can really render the speech incomprehensible, compare the above word with ***dha.Dakane*** [धड़कने, dhə.ɽək.ne, verb. To beat (heart), with *case-ending* ***ne***], where *schwa* following ***k*** is deleted. Without any schwa deletion, not only will the two words sound very unnatural, but it will also be extremely difficult for the listener to distinguish between the two.

The pronunciation of schwa is also context dependent in Hindi. For example in the word ***alaga*** [अलग, ʌ.ləg, separate] the pronunciation of the first and second schwas are different. The pronunciation of *anusvara* and chandra-bindu are also ambiguous in Hindi. For example, the anusvara may be pronounced as ***n***, ***N***, ***m*** or ***~N*** (both voiced and unvoiced) depending on the context. Sometimes the anusvara only stands for nasalization of a vowel and not a nasal consonant like in ***chiMTi*** [चींटी, ant].

Most of the existing Hindi *TTS* systems either neglect these issues and generate a flat, unnatural and sometimes unintelligible speech by retaining all the schwas, or they go for a dictionary based approach, which is expensive from storage point of view. Although the problems of *schwa deletion* and syllable boundary determination in Hindi (and other Indian Languages as well) have been addressed from a linguistic perspective [**7**], very little work has been done on the computational aspects. The only computational model for *schwa deletion* in Hindi that could be located is by B. Narsimhan et al [**10**]. Their work combines Ohala's work (1983) and *morphological analysis* with *finite state transducers* [**11**,**12**] and cost models achieving an accuracy of 89%.

This paper presents a schwa deletion algorithm for Hindi which achieves more than 96% accuracy. An extended algorithm, that uses word-morphology information, is also discussed, which boosts up the performance to 99.89%. Section 4 shows how the schwa deletion algorithm so designed can be used for syllable boundary marking using some very simple rules. The problem of *anusvara* and *chandra-bindu* pronunciation is dealt with next. The rules for context dependent pronunciation of schwa have not been included in this paper. However, it is to be noted that these nuances does not affect the

---

[2] The first vowel of Hindi alphabet, अ (pronounced as ə or ʌ, but for our convenience we shall denote it as ə only for both the contexts).

intelligibility of the speech and can be handled by proper intonation modeling of the language. There are other issues like pronunciation of numerals and acronyms. The rules for these are well known and will not be discussed here.

3. Algorithm for schwa deletion

First, some of the contexts where schwa is always retained or deleted have been identified and discussed, followed by the formal description of the algorithm, which makes use of these contexts. Most of these contexts arise due to phonotactic constraints and some due to syllable structure of Hindi.

1. The schwa of a syllable immediately followed by a conjugate syllable (*yuktakshara*) is always retained. For example in ***sAphalya*** [साफल्य, sa.ɸəl.jə, success] and ***AmantraNa*** [आमंत्रण, a.mən.trəɳ, invitation] the *schwas* following ***ph*** and ***m*** are retained.

2. If ***y*** (य) is followed by the inherent *schwa* and preceded by a syllable with a high vowel such as ***i***, ***I***, ***R^i, u*** or ***U,*** then the schwa following ***y*** is always retained. For example in ***priya*** [प्रिय, pri.jə, beloved]. On the other hand for low and medium height vowels like ***a***, ***A***, ***e*** or ***o***, the *schwa* following ***y*** may be deleted. For example in ***Aya*** [आय, aĕ, income].

3. Any conjugate syllable or cluster of consonants[3] that ends in (i.e. the last consonant of the cluster/syllable is) ***y***, ***r***, ***l*** or ***v***, the *schwa* following the cluster is retained. For example in ***kAvya*** [काव्य, kaw.jə, poetry], ***samprati*** [सम्प्रति, səm.prə.ti, recently], ***ashva*** [अश्व, əʃ.wə, horse] and shukla [शुक्ल, ʃuk.lə, white] the *schwas* following ***y***, ***r***, ***l*** and ***v*** are retained.

4. The schwa preceding a full vowel[4] is retained to maintain lexical distinctions. For example in the word ***ba.DhaI*** [बढ़ई, bə.ɽhəiː, carpenter] the schwa following ***.Dh*** is retained.

5. The *schwa* of the first syllable is never deleted. For example, the *schwas* following ***b*** in ***badarA*** [बदरा, bəd.ra, cloud], ***k*** in ***kalama*** [क्लम, kə.ləm, pen] or ***shh*** in ***kShamatA*** [क्षमता, kʂəm.ta, ability] are retained.

6. If the last syllable of the word contains a *schwa* and contexts 1 through 5 described above for the retention of the *schwa* do not occur, then the *schwa* is to be deleted. For example, the *schwas* following ***m*** in ***kalama, d*** in ***banda*** [बंद, bənd, closed] or ***k*** in ***tarka*** [तर्क, tərk, argument] are deleted.

---

[3] In Hindi, there can be a cluster of at most three consonants.

[4] Vowels can occur in two forms – full or *maatraas*. E.g. in the word ***AnA*** [आना, a.na, to come], the first ***A*** is a full vowel whereas the second one is a *mAtrA*

Whenever the above contexts arise, we can determine whether the schwa is to be retained or deleted. However, if none of the context arises, we cannot conclude anything.

3.1 The Algorithm

For description of the algorithm, we shall take the help of a notation called *half* ($\mathcal{H}$) and *full* ($\mathcal{F}$) sounds. We define a *full* sound as a consonant-vowel pair or a vowel alone, whereas *half* sound as a pure consonant sound, without any vowel modulation (*mAtrA*). Therefore, any vowel or a consonant followed by a vowel (*mAtrA*) is a *full* sound, whereas a consonant followed by *halant* (i.e. the consonants of a conjugate syllable or cluster, except the last one) are *half* sounds. Since mark (half or full) of the consonants followed by *schwa* might not be known beforehand, we shall call such consonants as *unknown* ($\mathcal{U}$). After marking the consonants of the word according to the rules stated above, only the consonants followed by *schwas* can be marked as $\mathcal{U}$. The algorithm then scans the marked word from left to right replacing each of the $\mathcal{U}$s by either $\mathcal{F}$ or $\mathcal{H}$, depending on the two adjacent syllables[5] of that particular $\mathcal{U}$-marked consonant. The basic idea here is to minimize the number of syllables in the word by deleting as many schwas as possible without violating any *phonotactic constraints*, which requires retention of those schwas which have an $\mathcal{H}$-marked consonant as at least one of its neighbors. At the end of the algorithm, schwas following the consonants marked as $\mathcal{H}$ are deleted.

The formal steps of the algorithm are described next. Figure 1 illustrates the stepwise execution of the algorithm on the words ***bachapana*** [बचपन, bəc.pən, childhood], ***priyatama*** [प्रियतम, pri.jə.təm, beloved] and ***AmantraNa***.

procedure *delete_schwa(DS)*

Input: word: string of alphabets (graphemes)[6]
Output: input word with some of the schwas deleted.

1. Mark all the full vowels and consonants followed by vowels other than the inherent *schwas* (i.e. consonants with *mAtrAs*) in the word as $\mathcal{F}$. Mark all the consonants immediately followed by consonants or *halants* (i.e. consonants of conjugate syllables) as $\mathcal{H}$. Mark all the remaining consonants, which are followed by implicit schwas as $\mathcal{U}$.
2. If in the word, ***y*** is marked $\mathcal{U}$ and preceded by ***i***, ***I, ri***, ***u*** or ***U*** mark it $\mathcal{F}$ (context 2).
3. If ***y***, ***r***, ***l*** or ***v*** are marked $\mathcal{U}$ and preceded by consonants marked $\mathcal{H}$, then mark them $\mathcal{F}$ (context 3).

[5] Note that here syllable refers to *akshhara* which can be consonant, vowel or a consonant vowel pair. In section 4 the syllable refers to speech units, which may or may not map to *akshharas*

[6] In order to keep the description of the algorithm simpler, the output is also presented as a string of graphemes instead of phonemes. After *schwa deletion*, grapheme to phoneme mapping for Hindi becomes quite simple.

4. If a consonant marked $\mathcal{U}$ is followed by a full vowel, then mark that consonant as $\mathcal{F}$ (context 4).
5. While traversing the word from left to right, if a consonant marked $\mathcal{U}$ is encountered before any consonant or vowel marked $\mathcal{F}$, then mark that consonant as $\mathcal{F}$ (context 5).
6. If the last consonant is marked $\mathcal{U}$, mark it $\mathcal{H}$ (context 6).
7. If any consonant marked $\mathcal{U}$ is immediately followed by a consonant marked $\mathcal{H}$, mark it $\mathcal{F}$ (context 1).
8. While traversing the word from left to right, for every consonant marked $\mathcal{U}$, mark it $\mathcal{H}$ if it is preceded by $\mathcal{F}$ and followed by $\mathcal{F}$ or $\mathcal{U}$ otherwise mark it $\mathcal{F}$.
9. For all consonants marked $\mathcal{H}$, if it is followed by a schwa in the original word, then delete the schwa from the word. The resulting new word is the required output.

end procedure *delete_schwa*

Word	***ba-cha-pa-na***	***p-ri-ya-ta-ma***	***A-ma-n-t-ra-Na***
After Step			
1	$\mathcal{U}$--$\mathcal{U}$--$\mathcal{U}$--$\mathcal{U}$	$\mathcal{H}$-$\mathcal{F}$-$\mathcal{U}$-$\mathcal{U}$-$\mathcal{U}$	$\mathcal{F}$-$\mathcal{U}$-$\mathcal{H}$-$\mathcal{H}$-$\mathcal{U}$-$\mathcal{U}$
2	$\mathcal{U}$--$\mathcal{U}$--$\mathcal{U}$--$\mathcal{U}$	$\mathcal{H}$-$\mathcal{F}$-$\mathcal{F}$-$\mathcal{U}$-$\mathcal{U}$	$\mathcal{F}$-$\mathcal{U}$-$\mathcal{H}$-$\mathcal{H}$-$\mathcal{U}$-$\mathcal{U}$
3	$\mathcal{U}$--$\mathcal{U}$--$\mathcal{U}$--$\mathcal{U}$	$\mathcal{H}$-$\mathcal{F}$-$\mathcal{F}$-$\mathcal{U}$-$\mathcal{U}$	$\mathcal{F}$-$\mathcal{U}$-$\mathcal{H}$-$\mathcal{H}$-$\mathcal{F}$-$\mathcal{U}$
4	$\mathcal{F}$--$\mathcal{U}$--$\mathcal{U}$--$\mathcal{U}$	$\mathcal{H}$-$\mathcal{F}$-$\mathcal{F}$-$\mathcal{U}$-$\mathcal{U}$	$\mathcal{F}$-$\mathcal{U}$-$\mathcal{H}$-$\mathcal{H}$-$\mathcal{F}$-$\mathcal{U}$
5	$\mathcal{F}$--$\mathcal{U}$--$\mathcal{U}$--$\mathcal{U}$	$\mathcal{H}$-$\mathcal{F}$-$\mathcal{F}$-$\mathcal{U}$-$\mathcal{U}$	$\mathcal{F}$-$\mathcal{U}$-$\mathcal{H}$-$\mathcal{H}$-$\mathcal{F}$-$\mathcal{U}$
6	$\mathcal{F}$--$\mathcal{U}$--$\mathcal{U}$--$\mathcal{H}$	$\mathcal{H}$-$\mathcal{F}$-$\mathcal{F}$-$\mathcal{U}$-$\mathcal{H}$	$\mathcal{F}$-$\mathcal{U}$-$\mathcal{H}$-$\mathcal{H}$-$\mathcal{F}$-$\mathcal{H}$
7	$\mathcal{F}$--$\mathcal{U}$--$\mathcal{U}$--$\mathcal{H}$	$\mathcal{H}$-$\mathcal{F}$-$\mathcal{F}$-$\mathcal{U}$-$\mathcal{H}$	$\mathcal{F}$-$\mathcal{F}$-$\mathcal{H}$-$\mathcal{H}$-$\mathcal{F}$-$\mathcal{H}$
8.1	$\mathcal{F}$--$\mathcal{H}$--$\mathcal{U}$--$\mathcal{H}$	$\mathcal{H}$-$\mathcal{F}$-$\mathcal{F}$-$\mathcal{F}$-$\mathcal{H}$	--
8.2	$\mathcal{F}$--$\mathcal{H}$--$\mathcal{F}$--$\mathcal{H}$	--	--
Results: 9	*bach-pan*	*pri-ya-tam*	*A-man-traN*

Figure 1 Illustration of the working of the algorithm

3.2 Morphological Analysis

The algorithm *DS* may produce erroneous results for *non-monomorphemic* words. For words made up of more than one morpheme as in the case of compound words, words with affixes or inflected forms, there is a tendency to retain the sounds of the morphemes. Therefore, given any word, it must be decomposed into stems and affixes and the algorithm *DS* is to be applied to each of the morphemes individually after which there pronunciations can be merged by following specific rules (not described here due to maintain brevity of the paper). For example, ***charaNakamala*** [चरणकमल, cə.rəɳ.kə.məl] => (after *morphological analysis*) ***charaNa*** [foot] & ***kamala*** [lotus] => (after individual *schwa deletion*) *cha-raN* & *ka-mal* => (after juxtaposition, final result) *cha-raN-ka-mal*.

(On the other hand, without *morphological analysis*, the result would have been *char-Nak-mal*, which is wrong.)

3.3 Performance

The algorithm *DS* has been implemented in C++ and integrated with *iLEAP*, a software supporting Indian language fonts. All the Hindi words in a pocket dictionary [**6**] were tested for *schwa deletion*. The output was checked manually. The results of the experiments are as follows.

Without *Morphological Analysis*:
Total number of words tested: 11095
Number of words with wrong *schwa deletion* results: 431
Thus, correctness of the algorithm: 96.12%
With a *Morphological Analyzer*:
Total number of words tested: 11095
Number of words with wrong *schwa deletion* results: 12
Thus, correctness of the algorithm: 99.89%

4. Syllable Boundary Marking

Syllable boundaries in Hindi also depend on the morpheme boundaries. Except for suffixes starting with a vowel (like ***Ina*** in ***namakIna*** [नमकीन, nəm.kiːn ,salty]), syllable breaks are always present at the morpheme boundaries. Within the morphemes, after schwa deletion, the syllable breaks can be determined by using the following rules.

.

1. Mark the consonants and full vowels as $\mathcal{F}$ or $\mathcal{H}$ as described by the algorithm *DS*.
2. Introduce syllable breaks between two consecutive $\mathcal{F}$s unless the second $\mathcal{F}$ corresponds to a full vowel. (E.g. *gA-nA* [7][गाना, song] but *ka-lAI* [क्लाई, wrist]).
3. Introduce syllable breaks between $\mathcal{H}$ and $\mathcal{F}$ unless the $\mathcal{H}$ corresponds to the first consonant of the word or is the beginning of a syllable.(E.g. *jan-tA* [जनता, public] but *pra-hAr*)
4. For conjugate syllable of two or three consonants, break is introduced after the first consonant. The other one or two consonants form the parts of the next syllable. However, this rule is not valid if the conjugate syllable is the last in the word.(E.g. *mak-khI* [मक्खी, fly], *sam-prati* but *band*).

These rules are sufficient for determining all the syllable boundaries in a word, after its morphological decomposition is known.

5. Anusvara and Chandra-bindu

In general, *anusvara* is pronounced as the nasal sound of the *varga* (class based on place of articulation) of the following consonant. Thus, in ***aMga*** [अंग, part] it is ***~N*** since *M* is the followed by ***g*** and ***~N*** is the nasal sound of that *varga*. Similarly, in ***aMchala*** [अंचल, location] it is ***~n***, in ***aMDA*** [अंडा, egg] it is ***N*** and in ***aMta*** [अंत, end] it is

[7] – indicates syllable breaks.

n and in ***aMbara*** [अंबर, sky]it is ***m***. When followed by other consonants (not belonging to any *varga*) it is pronounced as ***.m*** as in ***aMsha*** [अंश, part]. Since the sound of ***~n*** is identical to that of ***n***, they can be pronounced using the same phoneme. Whenever *anusvara* occurs as the last consonant of the word, it no longer remains a nasal sound; instead it nasalizes the vowel it follows as in the case of ***meM*** [में, in] and ***haiM*** [हैं, are]. The word ***ahaM*** [अहं, ego or self] is an exception to this rule. These two rules have been captured by a finite state transducer shown in Figure 2.

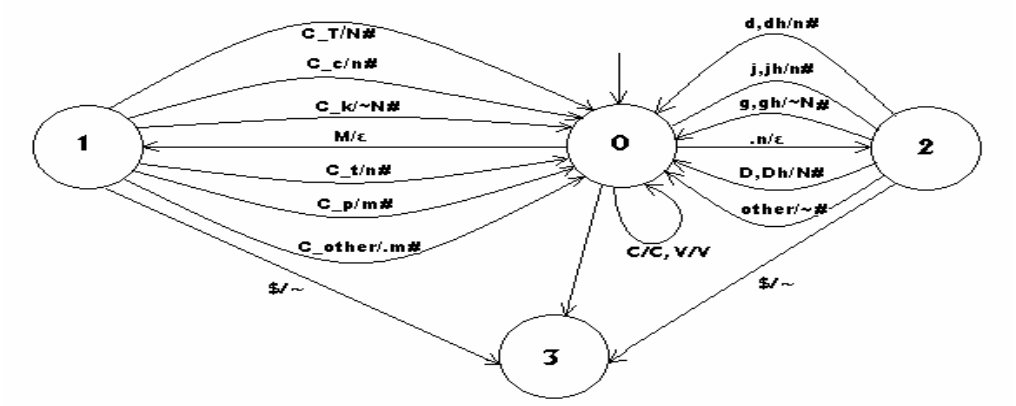

Figure 2: Finite state transducer for anusvara and chandra-bindu pronunciation rules. ~ represents nasalization of a vowel, C, V represents consonant and vowel. C_k denotes consonants of k varga which includes ***k***, ***kh***, ***g***,***gh***, ***q***, ***K*** and ***G***. Similarly other symbols are to be interpreted. # represents the input and .m stands for *anusvara* that follows sibilants.

In some contexts, even if *anusvara* is followed by some consonant, it is not pronounced as a nasal sound; rather it nasalizes the preceding vowel, like in ***choMcha*** (चोंच, cõc beak) or ***chIMTI***. In the current implementation, these are being treated as exceptions. *Chandra-bindu* is always used for nasalization of the vowel it follows unless it is followed by some voiced consonant, in which case it may be pronounced as a nasal sound also. In Hindi, many a times the same word is written interchangeably using *anusvara* and *chandra-bindu* like ***aMgura*** [अंगुर, grape] and ***a.Ngura*** [अँगुर]. On the other hand, the same word may have different pronunciations like ***sAMjha*** [सांझ, evening] (***M*** is pronounced both as ***n*** or nasalized ***A***). These facts aggravate the problem of correct pronunciation of *anusvara* and *chandra-bindu*. In the current model, the words which do not conform to the above rules are treated as exceptions. One possible approach is development of a statistical model for this problem.

6. Conclusion

In this paper, a computational framework for rule based grapheme to phoneme mapping for Hindi has been described. The system has been implemented in C++. Initially, the system converts consonants like ***j~n*** and vowels like ***r^i***, ***ai*** and ***au*** to the proper phonemes. Then *anusvara* and *chandra-bindu* are handled using the rules stated in section 5. The schwa deletion algorithm is applied to the word after that. In the process,

morphological decomposition of the word is also obtained. Finally, the syllable boundaries are marked using the schwa deletion algorithm. The rules have been implemented in form of finite state transducers. The system is being used by the concatenative speech synthesizer developed in Media Lab Asia, IIT Kharagpur.

The exceptions to the rules can be handled by exhaustive listing, as is done in the current system. The author is trying to develop a statistical model for covering those exceptions. Author is currently involved in design of Bengali grapheme to phoneme converter.

References

[1] Dutoit T., "An Introduction to Text-To-Speech Synthesis", *Kluwer Academic Publishers*, 1996.

[2] Allen J., Hunnicut S., Klatt D., "From Text To Speech, The MITTALK System", *Cambridge University Press*, 1987.

[3] Hunnicut S., "Grapheme-to-Phoneme rules: a Review", Speech Transmission Laboratory, Royal Institute of Technology, Stockholm, Sweden, QPSR 2-3, pp. 38-60.

[4] Belrhari R., Auberge V., Boe L.J., "From lexicon to rules: towards a descriptive method of French text-to-phonetics transcription", *Proc. ICSLP 92,* Alberta, pp. 1183-1186.

[5] Kager R., "Optimality Theory", *Cambridge University Press*, 1999

[6] "Hindi Bangla English – Tribhasa Abhidhaan", Sandhya Publication, 1st Edition March 2001

[7] Kaira S., "Schwa-deletion in Hindi", *Language forum (back volumes), Bhari publications*, Vol. 2, No. 1, April-June 1976

[8] Hooper J., "Constraints on schwa-deletion in American English", In *Recent Developments in Historical Linguistics*, Ed. By J. Fisiak, The Hague: Mouton, 1978 pp. 183-207

[9] Travel, Bernard, "Optional Schwa Deletion: on syllable economy in French", *Formal Perspectives on Romance Linguistics*, Ed. By J. Mark Authier, Barbar S. Bullock, & Lisa A. Reed., 1999

[10] Narasimhan B., Sproat R., and Kiraz G., "Schwa-deletion in Hindi Text-to-Speech Synthesis," *Workshop on Computational Linguistics in South Asian Languages*, 21st SALA, October 2001, Konstanz

[11] Kaplan R. M., Kay M., "Regular models of phonological rule systems", *Computational Linguistics*, Vol. 20, no. 3, pp. 331-378, Sept. 1994

[12] Mohri M., Sproat R., "An efficient compiler for weighted rewrite rules" in *Proceedings of 34th Mtg of the Association for Computational Linguistics*, Santa Cruz, June 1996, pp. 231-238