On Decidability of Sanskrit Morphophonological Processes

NIKHIL CHATURVEDI, Indian Institute of Technology, Delhi, India
SHUBHAM BHARDWAJ, Indian Institute of Technology, Delhi, India
RAHUL GARG, Indian Institute of Technology, Delhi, India
PRATHOSH A. P., Indian Institute of Technology, Delhi, India

The Sanskrit language has a precise and completely specified grammar given in the text *aṣṭādhyāyī* by *Pāṇini*. One of the most basic morphophonological process in Sanskrit is that of Sandhi, which pertains to modification of phonemes along closely knit morpheme or word boundaries. The underlying essence of Sandhi, i.e., modification of phonemes, shares its presence in numerous modern languages like Hindi, English, French and even Mandarin. In this paper we analyze the computational complexity of morphophonological processes which involve basic letter-level modifications of *elision*, *augmentation* and *substitution* in settings with arbitrary rule sets. We formulate the above using notions of a formal Sandhi Grammar and deduce its undecidability in a unrestricted setting. We further demonstrate that under specific restrictions, the proposed Sandhi Grammar becomes decidable. We also examine the problem of Sanskrit Sandhi and show its decidability in certain restricted settings.

1 INTRODUCTION

1.1 Background

Sanskrit is one of the oldest members of the proto-Indo-European family of languages, being the earliest of the Indic languages [Fortson IV 2011]. The body of literature in Sanskrit language is broadly classified into two categories namely, vedic and classical Sanskrit [Keith 1993]. Vedic Sanskrit constitutes a large set of mystic hymns known to be the oldest surviving text of the Sanskrit language [Thieme 1935]. Classical Sanskrit encompasses the entire gamut of non-vedic literature including Smritis, Itihasas, Kavyas and theological and philosophical texts [Dasgupta 1966].

One of the striking features of the Sanskrit language is an accurate codification of its grammar rules. *Pāṇini*, an ancient grammarian constructed a set of 3959 aphoristic statemets in his famous book the *Aṣṭādhyāyī* [Katre et al. 1989; SD 2017] to explain the phonological and morphological formations of words in both Vedic and classical texts [Staal 1965], using a genertive grammatical structure. The rules laid down in *aṣṭādhyāyī* are precise and cover a large body of available text in the Sanskrit language. Since most of the authors post *Pāṇini* have adherered to the rules of *aṣṭādhyāyī*, the language has not gone through much modification through time. Such a specification of the language has encouraged numerous grammarians in the field of Sanskrit Computational Linguistics to develop tools for assisting students in the language and for translation and interpretation of the text by general readers. For instance, the group from INRIA [Gerard 2018], has developed a number of tools such as lemmatizer and morphological analyzer. Another significant effort towards this direction is by the group at JNU [Jha 2017] that has a range of tools from PoS tagger to morphological decomposers. The final aim of all these tools is to develop a fully-functional Sanskrit parser albeit they comprise of sub-tools such as dictionaries, transliterators, morphological splitters,

Authors' addresses: Nikhil Chaturvedi, Computer Science, Indian Institute of Technology, Delhi, Hauz Khas, Delhi, 110016, India, 709nikhil@gmail.com; Shubham Bhardwaj, Electrical Engineering, Indian Institute of Technology, Delhi, Hauz Khas, Delhi, 110016, India, shubhamiitd.007@gmail.com; Rahul Garg, Computer Science, Indian Institute of Technology, Delhi, Hauz Khas, Delhi, 110016, India, rahulgarg@cse.iitd.ac.in; Prathosh A. P., Electrical Engineering, Indian Institute of Technology, Delhi, Hauz Khas, Delhi, 110016, India, prathoshap@ee.iitd.ac.in.

2018. 2475-1421/2018/1-ART1 $15.00
https://doi.org/

and generators and so on. Once realized fully, these tools aid a non-expert to access the content (such as Yogic texts) in their original form.

1.2 Scope and Objectives

Even though there have been multiple attempts to create automated ad-hoc tools for morphological processess in *Aṣṭādhyāyī*, there is very less work on formalization any of these notions. Further, a recent study created a benchmark corpus [Shubham et al. 2018] for one such set of morphological process called the Sandhi and demonstrated that several of exisiting tools such as [Goyal and Huet 2013; Jha 2017; Kulkarni 2017] do not perform well on such task. Thus, owing to the need for a robust tool, in this paper, we formalize one class of grammatical process called the Sandhi that broadly refers to the morphophonological deformations that pairs of consecutive words go through in continuous speech (Precise definition and examples given in Section 2). Specifically, we formulate of the problem of Sandhi using the notions of formal grammar and propose three methods of application of such grammar to the Sandhi problem. We prove that for the unconstrained variants of the above grammar, the Sandhi problem is undecidable for all three methods. Subsequently, we deduce the decidability of proposed methods under specific constraints. Next, we show that the problem of Sanskrit Sandhi adheres to such constraints and thus is decidable. Finally, we validate and compare our proposed formulation with existing Sandhi tools.

2 MORPHONOLOGICAL PROCESSES IN SANSKRIT

2.1 The Process of Sandhi

Word formation in Sanskrit is centered around a root verb, modified by a suffix (and additionally a prefix in certain cases). Each of these three (roots, prefixes, and suffixes) represents a morpheme category, as these are the meaningful morphological units of the language and none of them can be further divided. Sanskrit texts often contain words which are formed by the combination of two or more words. The process of combining two (or in certain cases more) consecutive words (or morphemes) during continuous speech is known as Sandhi. The reverse process of getting back the component morphemes/words from such words is known as Sandhi Viccheda or Sandhi splitting. *Aṣṭādhyāyī* codifies the Sandhi rules in 272 aphorisms, referred to as *saṃhita*, which is the phonological modification of obtained due to close proximity. For example, when the words *sūrya* and *udaya* are combined, we get the dipthong *'o'* which represents a combination of *'a'* and *'u'* in terms of sound. Thus, the words end up merging into *sūryodaya*, a result which can intuitively be reproduced by repeating the words *sūrya* and *udaya* quickly one after the other. The process of Sandhi between two words is not always determined only by the two combining sounds (represented by the last letter of the first word and the first letter of the second word) but may also depend on the letters present elsewhere in either of the words, grammatical aspects of the two words (the parts of speech they belong to, declensions, conjugations, etc), their meaning, whether they occur within verses or not or even a combination of one or more of these.

However, for the purpose of this paper, we have focused only on the most prominent aspect of Sandhi where two consecutive sounds (typically the last letter of the first word and the first letter of the next word in sequence) get modified according to a well-defined set of morphophonological rules, irrespective of the broader context in which they appear. For example, the rule that leads to *sūryodaya* (meaning sunrise) from *sūrya* (sun) and *udaya* (rise), (Rule 6.1.87 of *aṣṭādhyāyī*) takes into consideration only the combining two letters. According to this rule, if the first word ends with an *a/ā* and the second word starts with *i/ī* or *u/ū*, the two letters combine to give *e/o* respectively. No other information regarding the structure of the two words, the grammatical categories, semantics, etc. is required.

The rules pertaining to Sandhi formation described in *Aṣṭādhyāyī* can be categorized into following broad categories -Augmentation, Substitution, Elision, Combination and Concatenation. Further, Augmentation, Substitution and Elision can occur on both the first as well as the second word whereas Combination and Concatenation occur only in the second word. Thus, Sandhi considered in this study can be classified into eight following categories:

- **Pre-Augmentation:** Adds a letter after the last letter of the first word
- **Post-Augmentation:** Adds a letter before the first letter of the second word
- **Pre-Substitution:** Substitutes the last letter of the first word with another
- **Post-Substitution:** Substitutes the first letter of the second word with another
- **Pre-Elision:** Removes the last letter of the first word
- **Post-Elision:** Removes the first letter of the second word
- **Concatenation:** Conjoining the two words into one without any change
- **Combination:** Combining the last letter of the first word and the first letter of the last word into a single letter

In the above described types, all except Combination and Concatenation leave scope of further Sandhi based on other rules, thus making the Sandhi process recursive in nature. The existence of such a recursion opens up discussion about the possibility and/or conditions for termination of the Sandhi of any pair of given words. This tractability analysis forms the motivation for the subsequent sections.

2.2 Sandhi Merging and Splitting

Sandhi is a very common occurrence in Sanskrit that is a much more involved process as compared to similar processes in other languages. For example, in a language like English, the meaning of the words and parts of speech involved determine whether the words can be combined or not. Thus, for example, in the sentence 'The regrettable decision of the chairman is now causing great harm to him', each of the words 'regrettable' , 'chairman' and 'causing' represents a combination, but combinations like 'Theregrettable' or 'regrettabledecision' or 'isnow' or worse 'Theregrettabledecisionofthechairmanisnowcausinggreatharmtohim' are simply not allowed.

On the other hand, in Sanskrit, these are not only allowed but encountered very frequently. So while no combination is possible between the words of the sentence 'Ravi arrived in forest', all the words in the Sanskrit equivalent *raviḥ vane āgataḥ* can combine to form *ravirvanayāgataḥ* (please note the changes at the boundaries of merging). Thus, Sandhi splitting, which decomposes the complex compound like *ravirvanayāgataḥ* into its constituent elements *raviḥ vane āgataḥ*, is an indispensable first-step in the analysis of classical Sanskrit texts.

Unlike Sandhi merging where the word boundary between two given words is well defined, in the case of Sandhi splitting not only is the split location in the single word input unknown, but the single word may be composed of more than two syntactically valid components. Moreover, there are cases where splitting at different locations may lead to different split combinations, each of which is not only syntactically valid, but also semantically valid. For example, the word *hasannāgachati* can either be split into *hasan* + *āgachati* (comes laughing) or *hasan* + *na* +*āgachati* (comes not laughing) both of which are not only syntactically but also semantically valid (to the extent of having opposite meanings!)

Thus, based on these observations, the Sandhi process can be categorized into two types - (i) modification that focuses across a well-defined word boundary created by a given pair of distinct words. We call this type of the merging process as **Definite Location Sandhi Merging**. (ii) Modifications that functions on arbitrary locations within a word. The word boundary in this case can correspond to any pair of consecutive letters within the word. We term this type of merging

process **Arbitrary Location Sandhi Merging/Splitting**. In the subsequent sections, we follow the notations and definitions given in this section to formulate the Sandhi creation and splitting problems as a formal grammar and discuss its decidability under different conditions.

3 SANDHI GRAMMAR (MORPHOPHONOLOGICAL GRAMMAR)

This section begins with a formal definition of *Sandhi Grammar* followed by an example toy grammar motivated by morphophonological processes in the English language.

Definition 1 (Sandhi Grammar). *A Sandhi Grammar is defined as* $\mathcal{G} = (\Sigma, \mathcal{R})$ *where* Σ *is a finite set of alphabets of the language,* $\mathcal{R}$ *is a (possibly) multi-valued partial function* $\mathcal{R} : \Sigma \times \Sigma \longmapsto \mathcal{T} \times \Sigma$ *representing the morphophonological processes in the language (also called rules of the Sandhi Grammar), and* $\mathcal{T}$ *represents the set of 6 possible morphophonological modifications (Sandhi Types),* $\mathcal{T}$ *= {Pre-Augmentation, Post-Augmentation, Pre-Substitution, Post-Substitution, Pre-Elision, Post-Elision}.*

A *Sandhi Grammar* may be used to formalize morphophonological processes that modify consecutive sounds in a language in a manner that is independent of wider context in which these sounds occur. Such processes may be deterministic or non-deterministic in natural languages. The definition above takes care of both these possibilities.

Definition 2 (Deterministic Sandhi Grammar). *A Sandhi Grammar is deterministic if* $\mathcal{R}$ *is a single-valued function, and non-deterministic otherwise.*

A single modification of adjacent sounds based on the rules of the grammar is formalized as a *Sandhi Operation.*

Definition 3 (Sandhi Operation). *Given a Sandhi Grammar* $\mathcal{G}$*, the Sandhi Operation* $\Phi_{\mathcal{G}}$ *is a multi-valued function that takes two letters as input and produces strings of length 1, 2 or 3 as output by using the rules of the Sandhi Grammar.* $\Phi : \Sigma \times \Sigma \longmapsto \Sigma^1 \cup \Sigma^2 \cup \Sigma^3$.

$$
\begin{aligned}
&\textit{Let } (\tau, c) \in \mathcal{R}(c_1, c_2), \\
&\qquad \Phi_{\mathcal{G}}(c_1, c_2) \ni c_1 c c_2 && \textit{if } \tau = \textit{Pre-Augmentation} \\
&\qquad\qquad \ni c_1 c c_2 && \textit{if } \tau = \textit{Post-Augmentation} \\
&\qquad\qquad \ni c c_2 && \textit{if } \tau = \textit{Pre-Substitution} \\
&\qquad\qquad \ni c_1 c && \textit{if } \tau = \textit{Post-Substitution} \\
&\qquad\qquad \ni c_2 && \textit{if } \tau = \textit{Pre-Elision} \\
&\qquad\qquad \ni c_1 && \textit{if } \tau = \textit{Post-Elision}
\end{aligned}
$$

Remark. The Sandhi Operation Φ is a partially defined multi-valued function. $\Phi_{\mathcal{G}}(c_1, c_2) = \phi$ if $\mathcal{R}(c_1, c_2) = \phi$, where ϕ denotes the null set.

Remark. For a deterministic Sandhi Grammar, the Sandhi Operation $\Phi_{\mathcal{G}}$ is single-valued (or null), whereas for a non-deterministic Grammar it is multi-valued.

Remark. For $(\tau, c) \in \mathcal{R}(c_1, c_2)$ if $\tau \in \{\text{Pre-Elision}, \text{Post-Elision}\}$ the letter c is ignored and hence, for clarity, the special symbol ϵ will be used in case of elisions.

Given a string, the grammar may allow modification of sounds at multiple locations in the string. The locations where such modifications are possible according to the rules of the Sandhi grammar are formalized as applicable sandhi locations.

Definition 4 (Applicable Sandhi Location). *A location* κ *in the word* $w = c_1 c_2 \ldots c_l \in \Sigma^*$ $(l > \kappa)$ *is an Applicable Sandhi Location if* $\Phi(c_\kappa, c_{\kappa+1}) \neq \phi$.

Remark. If $\Phi(c_\kappa, c_{\kappa+1}) = \phi$ at the location κ ($\kappa < l$) in the word $w = c_1c_2 \dots c_l \in \Sigma^*$ or $\kappa \leq 0$ or $\kappa \geq l$, the Sandhi Location is not an Applicable Sandhi Location.

Example 1. *We now create an example Sandhi Grammar which we shall refer to throughout this paper. Through this Grammar, we aim to motivate certain morphophonological processes that take place in the English language. We describe this grammar as* $\mathcal{G}_E = (\Sigma_E, \mathcal{R}_E)$ *where* Σ_E = *The English Lowercase Alphabet, and* $\mathcal{R}_E$ *is defined in Table 1.*

No.	Input (c_1, c_2)	Output $(\mathcal{R}(c_1, c_2))$	Action
Rule 1	(y, e)	(Pre-Substitution, i)	ye $\longrightarrow$ ie
Rule 2	(y, i)	(Pre-Substitution, i)	yi $\longrightarrow$ ii
Rule 3	(f, i)	(Pre-Substitution, v)	fi $\longrightarrow$ vi
Rule 4	(d, i)	(Pre-Substitution, s)	di $\longrightarrow$ si
Rule 5	(e, e)	(Pre-Elision, ϵ)	ee $\longrightarrow$ e
Rule 6	(i, i)	(Pre-Elision, ϵ)	ii $\longrightarrow$ i
Rule 7*	(e, i)	(Pre-Elision, ϵ)	ei $\longrightarrow$ i
Rule 8*	(e, i)	(Post-Augmentation, t)	ei $\longrightarrow$ eti
Rule 9	(t, a)	(Pre-Augmentation, t)	ta $\longrightarrow$ tta

Table 1. Rules of an example Sandhi Grammar $\mathcal{G}_E$, motivated by morphophonological processes in the English Language

Remark. The rules marked with asterisk (*) make $\mathcal{G}_E$ non-deterministic.

Remark. Although the Sandhi Grammar $\mathcal{G}_E$ of Example 1 is motivated by a small subset of morphophonological processes in the English Language, its rules are not universally applicable. A more realistic Sandhi Grammar is that of the Sanskrit language where the corresponding rules have much more universal applicability. This Sandhi Grammar is listed in Appendix B.

Having formally defined the Sandhi Grammar, we now define various operations on strings that can be done using the rules of the grammar. These operations are *definite location sandhi merging* (DLSM), *arbitrary location sandhi merging* (ALSM) and *arbitrary location sandhi splitting* (ALSS).

4 DEFINITE LOCATION SANDHI MERGING (DLSM)

Definition 5 (DLSM Step)**.** *A Definite Location Sandhi Merging Step, at a given Applicable Sandhi Location* κ *in the word* $w = c_1c_2 \dots c_l \in \Sigma^*$ $(l > \kappa)$*, produces another word* $w' \in \Sigma^*$ *by application of one Sandhi Operation. If* $w = c_1c_2 \dots c_l$*, then* $w' = c_1 \dots c_{\kappa-1}sc_{\kappa+2} \dots c_l$*, where* $s \in \Phi_{\mathcal{G}}(c_\kappa, c_{\kappa+1})$*. Furthermore, it updates* κ *as follows:*

$$
\begin{aligned}
\kappa_{new} &= \kappa + 1 && \text{if } \tau = \textit{Pre-Augmentation} \\
&= \kappa && \text{if } \tau = \textit{Post-Augmentation} \\
&= \kappa && \text{if } \tau = \textit{Pre-Substitution} \\
&= \kappa && \text{if } \tau = \textit{Post-Substitution} \\
&= \kappa - 1 && \text{if } \tau = \textit{Pre-Elision} \\
&= \kappa && \text{if } \tau = \textit{Post-Elision} \\
& && \text{where } (\tau, c) \in \mathcal{R}(c_\kappa, c_{\kappa+1})
\end{aligned}
$$

Remark. If the given Sandhi Location κ is not an Applicable Sandhi Location then the DSLM Step is deemed as invalid.

Remark. The above definition is valid for both deterministic as well as non-deterministic Sandhi Grammars. In the case of a deterministic Sandhi Grammar the DLSM step produces a unique output, whereas in the case of a non-deterministic Grammar, the DLSM step is non-deterministic and produces multiple outputs depending on which rule is applied.

In general, the DLSM step may be applied multiple times on a string to generate the final string. This is formalized using the DLSM sequence.

Definition 6 (DLSM Sequence)**.** *A Definite Location Sandhi Merging Sequence of length n starting at Sandhi Location κ_0 in the word w ($|w| > \kappa_0$) produces another string w' with n applications of Sandhi Merging Steps starting at Sandhi Location κ_0 and updating the Sandhi Location as given in the Sandhi Merging Step.*

Example 2. *The following are the examples of DLSM steps and DLSM sequences for the grammar $\mathcal{G}_E$ of Example 1:*

- *happyer (happy + er)* $\xrightarrow{DLSM\ (\kappa=5)}$ *happier [Rule 1]*
- *driveing (drive + ing)* $\xrightarrow{DLSM\ (\kappa=5)}$ *driving [Rule 7]*
- *knifeing (knife + ing)* $\xrightarrow{DLSM\ (\kappa=5)}$ *knifing [Rule 7]* $\xrightarrow{DLSM\ (\kappa=4)}$ *kniving [Rule 3]*
- *erodeion (erode + ion)* $\xrightarrow{DLSM\ (\kappa=5)}$ *erodion [Rule 7]* $\xrightarrow{DLSM\ (\kappa=4)}$ *erosion [Rule 4]*

Remark. If the Sandhi Grammar is non-deterministic, multiple DLSM sequences are possible given an input w and a starting Sandhi Location κ_0 as long as no invalid DLSM Steps are taken.

Given a Sandhi Grammar and an input string and a starting Sandhi location, an important question is to determine if every DLSM sequence will converge i.e., lead to an output string where no more DLSM steps can be applied. This is called the DLSM termination problem.

Definition 7 (DLSM Termination Problem)**.** *Given an input Sandhi Grammar $\mathcal{G}$, word $w \in \Sigma^*$, and a starting Sandhi Location κ_0, the DLSM Termination Problem is to determine if $\exists n \in \mathbb{N}$ such that there exists no DLSM Sequence of length n.*

For real languages, it is important to design efficient algorithms that can produce all possible strings that can be obtained, starting with a given string, by repeated applications of DLSM steps. The set of all such strings is characterized by the *DLSM production set* and the corresponding membership problem by the *DLSM derivation problem.*

Definition 8 (DLSM Production Set)**.** *Given an input Sandhi Grammar $\mathcal{G}$, word $w \in \Sigma^*$, and a starting Sandhi Location κ_0, the DLSM Production Set $\Omega^{\mathcal{G}}_{DLSM}(w, \kappa_0)$ is defined as the set of strings w' such that there exists a DLSM Sequence of finite length which produces w' from w starting at Sandhi Location κ_0.*

Remark. If $\Omega^{\mathcal{G}}_{DLSM}(w, \kappa_0)$ is not finite, then the DSLM Sequence starting at κ_0 in the word w for the Sandhi Grammar $\mathcal{G}$ is non-terminating. Although its converse is not true, even if a DLSM Sequence is not terminating, the corresponding DLSM Production Set $\Omega^{\mathcal{G}}_{DLSM}(w, \kappa_0)$ may still be finite.

Definition 9 (DLSM Derivation Problem)**.** *Given an input Sandhi Grammar $\mathcal{G}$, a word $w_{in} \in \Sigma^*$, a starting Sandhi Location κ_0 and a word $w_{out} \in \Sigma^*$, the DLSM Derivation Problem is to determine if $w_{out} \in \Omega^{\mathcal{G}}_{DLSM}(w_{in}, \kappa_0)$.*

One of our important result is that for general Sandhi Grammars, the DLSM termination and DLSM derivation problems are undecidable.

Theorem 1. *The Definite Location Sandhi Merging Termination Problem is undecidable for a general Sandhi Grammar.*

The prof of Theorem 1 uses a general reduction from a Turing machine to the DLSM termination problem and is given in the Appendix. The undecidability of the DLSM derivation problem follows as a corollary of the reduction.

Corollary 1.1. *The Definite Location Sandhi Merging Derivation Problem is undecidable for a general Sandhi Grammar.*

The above results show that morphophonological processes that only modify consecutive letters (such as at the boundaries of word morphemes) and produce new words based on merely 6 types of rules can in general encode a Turing machine. This leads to the question about whether there are some natural languages where the morphophonological processes can become as complex as Turing Machines?

Most of the real languages do not have strictly rules-driven and well-defined morphophonological processes. However, Sanskrit does have well-defined and precise rules that completely characterize its grammar. A subset of 294 of those rules deal with the morphophonological process of Sandhi are given under Appendix. Thus arises the question of can Sanskrit Sandhi process encode a Turing Machine? More generally, what sub-types of Sandhi Grammars can or cannot encode a Turing Machine? Next definition motivates a class of Sandhi Grammars where the DLSM termination problem is decidable.

Definition 10 (Sandhi Transition Graph (STG))**.** *Given a Sandhi Grammar $\mathcal{G} = (\Sigma, \mathcal{R})$, we define the Sandhi Transition Graph as a directed multigraph $G = (V, E)$ where $V = \Sigma \times \Sigma$, and E is constructed as follows. $w(e)$ denotes weight of edge e.*

$$
\begin{aligned}
&\textit{Let } (\tau, c) \in \mathcal{R}(c_1, c_2), \\
&e = ((c_1, c_2), (c, c_2)) \in E, w(e) = +1 && \textit{if } \tau = \textit{Pre-Augmentation} \\
&e = ((c_1, c_2), (c_1, c)) \in E, w(e) = +1 && \textit{if } \tau = \textit{Post-Augmentation} \\
&e = ((c_1, c_2), (c, c_2)) \in E, w(e) = 0 && \textit{if } \tau = \textit{Pre-Substitution} \\
&e = ((c_1, c_2), (c_1, c)) \in E, w(e) = 0 && \textit{if } \tau = \textit{Post-Substitution} \\
&e = ((c_1, c_2), (c', c_2)) \in E, w(e) = -1 \ \forall c' \in \Sigma && \textit{if } \tau = \textit{Pre-Elision} \\
&e = ((c_1, c_2), (c_1, c')) \in E, w(e) = -1 \ \forall c' \in \Sigma && \textit{if } \tau = \textit{Post-Elision}
\end{aligned}
$$

Theorem 2. *If a Sandhi Transition Graph for a given Sandhi Grammar has no non-negative weight cycles, then the DSLM Sequence for every word w given any starting Sandhi Location κ_0 ($\kappa_0 < |w|$) terminates.*

The formal proof is given in the Appendix. The proof relies on the fact that since there are only a finite number of pairs of alphabets that can interact through a DLSM Sequence, non-termination will happen only when some of these pairs repeat. Cycles in the STG represent the possibility of such a repetition. Moreover if traversing one of these cycles reduces the overall length of the word w, then termination would happen eventually as the word length is finite and no valid Sandhi Locations will remain after $|w| = 1$.

Theorem 3. *For the morphophonological Sanskrit Grammar described in Appendix B, the DLSM Termination Problem is decidable.*

The Sanskrit Sandhi Grammar as formalized in the Appendix from the Sandhi rules of the *aṣṭādhyāyī*, does have a few non-negative weight cycles. However, these cycles are usually of small

in length and only a few such cycles exist. Therefore, by relying on Theorem 2 and then by carefully examining these small number of cases it can be shown that the DLSM termination problem is decidable for Sanskrit Sandhi Grammar. A more detailed proof summary is given in the Appendix.

5 ARBITRARY LOCATION SANDHI MERGING (ALSM)

Definite Location Sandhi Merging only covers a specific class of morphophonological processes which occur at the boundaries of two different words (external Sandhis in the case of Sanskrit language). However, morphophonological modifications can also occur within a word when the word gets modified using prefixes and suffixes (internal Sandhi). This gives rise to the *arbitrary location* Sandhi problems formalized next.

The *arbitrary location Sandhi merging step* is very similar to the DSLM step, except that the Sandhi location is not specified. Any applicable Sandhi location may be used for the ALSM step.

Definition 11 (ALSM Step)**.** *An Arbitrary Location Sandhi Merging Step, if there exists an applicable Sandhi Location κ in the word $w = c_1c_2 \dots c_l \in \Sigma^*$ $(\kappa < l)$, produces another word $w' \in \Sigma^*$ by application of one Sandhi Operation. If $w = c_1c_2 \dots c_l$, then $w' = c_1 \dots c_{\kappa-1}sc_{\kappa+2} \dots c_l$, where $s \in \Phi(c_\kappa, c_{\kappa+1})$.*

Remark. If no applicable Sandhi Location is present in w, then the ASLM Step is deemed as invalid.

Remark. The above definition supports both deterministic and non-deterministic Sandhi Grammars in a manner analogous to the DSLM step.

The *ALSM sequence* is defined analogously to the DSLM sequence, except that the Sandhi location is not pre-specified and it may be arbitrary.

Definition 12 (ALSM Sequence)**.** *An Arbitrary Location Sandhi Merging Sequence of length n for the word w produces another string w' with n applications of Sandhi Merging Steps.*

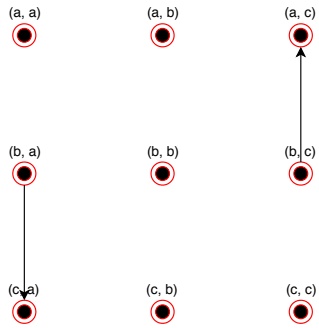
Remark. Unlike DLSM sequence, multiple ALSM sequences can be produced for a word w, even if the underlying Sandhi Grammar is deterministic, by using different applicable Sandhi locations.

Example 3. *The following example illustrates the ALSM steps in constructing the word demonetization from its constituents using the rules of the toy grammar $\mathcal{G}_E$ of Example 1:*

- *demoneyizeation (de + money + ize + ation) $\xrightarrow{ALSM}$ demoneiizeation [Rule 2] $\xrightarrow{ALSM}$ demoneizeation [Rule 6] $\xrightarrow{ALSM}$ demonetizeation [Rule 8] $\xrightarrow{ALSM}$ demonetization [Rule 7]*
- *demoneyizeation (de + money + ize + ation) $\xrightarrow{ALSM}$ demoneiizeation [Rule 2] $\xrightarrow{ALSM}$ demoneizeation [Rule 6] $\xrightarrow{ALSM}$ demonizeation [Rule 7] $\xrightarrow{ALSM}$ demonization [Rule 7]*

The ALSM termination problem is different from the DLSM termination problem. The result of Theorem 2 does not apply to the ALSM sequences. The following example illustrates that, unlike the DSLM sequence, the ALSM sequence for a grammar need not terminate even if the corresponding Sandhi transition graph has no cycles.

Example 4. *The Sandhi Grammar $\mathcal{G}_C$ has three alphabets and two rules defined as follows: $\mathcal{G}_C =< \{a, b, c\}, \mathcal{R} >$ where $\mathcal{R}(b, a) = \{(Pre\text{-}Augmentation, c)\}$ and $\mathcal{R}(b, c) = \{(Pre\text{-}Augmentation, a)\}$. The Sandhi transition graph for $\mathcal{G}_C$ has no cycles as depicted in the figure below.*



Clearly, the STG for $\mathcal{G}_C$ has no cycles, so by Theorem 2 all DLSM sequences given $\mathcal{G}_C$ will terminate regardless of the input. However, the same does not hold true for ALSM. Given an input string ba:

$$ba \xrightarrow{ba\to bca} bca \xrightarrow{bc\to bac} baca \xrightarrow{ba\to bca} bcaca \xrightarrow{bc\to bac} bacaca \ldots$$

As can be seen, even without any cycle in the STG, the ALSM sequence need not terminate on all inputs.

The *ALSM termination problem* is defined analogously to the DLSM termination problem.

Definition 13 (ALSM Termination Problem)**.** *Given an input Sandhi Grammar $\mathcal{G}$ and a word $w \in \Sigma^*$, the ALSM Termination Problem is to determine if $\exists n \in \mathbb{N}$ such that there exists no ALSM Sequence of length n.*

Remark. If no such n exists, then the ASLM Sequence for the word w given the Sandhi Grammar $\mathcal{G}$ is said to be non-terminating. In such a case, it is possible to find an arbitrarily large valid ALSM sequence leading to non-termination of ALSM steps.

Remark. The above definition is valid for both deterministic as well as non-deterministic Sandhi Grammars in a way analogous to the DLSM termination problem.

The *ALSM production set* and the *ALSM derivation problem* are defined analogus to the corresponding DLSM production set and the DLSM derivation problem.

Definition 14 (ALSM Production Set)**.** *Given an input Sandhi Grammar $\mathcal{G}$ and a word $w \in \Sigma^*$, the ALSM Production Set $\Omega^{\mathcal{G}}_{ALSM}(w)$ is defined as the set of strings w' such that there exists an ALSM Sequence of finite length which produces w' from w.*

Definition 15 (ALSM Derivation Problem)**.** *Given an input Sandhi Grammar $\mathcal{G}$, a word $w_{in} \in \Sigma^*$ and a word $w_{out} \in \Sigma^*$, the ALSM Derivation Problem is to determine if $w_{out} \in \Omega^{\mathcal{G}}_{ALSM}(w_{in})$.*

The following results can be directly obtained from Theorem 1, but are being given here for the sake of completeness.

Theorem 4. *The ALSM Termination Problem is undecidable for a general Sandhi Grammar.*

Corollary 4.1. *The ALSM Derivation Problem is undecidable for a general Sandhi Grammar.*

5.1 Specific Classes of ALSM Sub-problems

Unlike DLSM, absence of cycles in the Sandhi transition graph does not guarantee termination of ALSM sequences. A natural question arises about if there are some other properties of the Sandhi Grammar that may guarantee the decidability of the ALSM termination problem. In this subsection,

we give three seperate resctictions on the Sandhi grammar under which the ALSM problems are decidable.

The following result establishes that the ALSM termination problem is decidable in case the Sandhi grammar has no augmentation rules.

Theorem 5. *The ALSM Termination Problem is decidable for a Sandhi Grammar $\mathcal{G} = (\Sigma, \mathcal{R})$ where all rules in $\mathcal{R}$ follow a reduced set of Sandhi Types with no Pre-Augmentation or Post-Augmentation. $\mathcal{R} : \Sigma \times \Sigma \longmapsto \mathcal{T} \times \Sigma$, $\mathcal{T}$ = {Pre-Substitution, Post-Substitution, Pre-Elision, Post-Elision}.*

In the absence of augmentation rules, the ALSM step can never increase the length of the input string. The proof of the above theorem critically relies on this observation and constructs a graph on Σ^n where n is the length of the input string. The edges in the graph represent the ALSM steps. The ALSM sequence is non-terminating if there is a path from the node corresponding to the input to a cycle in the constructed graph. The formal proof is detailed in the Appendix.

The next result establishes decidability of the ALSM termination problem in the case of absence of elisions.

Theorem 6. *The Arbitrary Location Sandhi Merging Derivation Problem is decidable for a Sandhi Grammar $\mathcal{G} = (\Sigma, \mathcal{R})$ where all rules in $\mathcal{R}$ follow a reduced set of Sandhi Types with no Pre-Elision or Post-Elision. $\mathcal{R} : \Sigma \times \Sigma \longmapsto \mathcal{T} \times \Sigma$, $\mathcal{T}$ = {Pre-Substitution, Post-Substitution, Pre-Augmentation, Post-Augmentation}. Furthermore, such a Grammar can be represented by a Context-Sensitive Grammar.*

The proof follows from a reduction to the context sensitive grammars as detailed in the Appendix.

The augmentation, substitutions and elisions can interact in a complex manner to simulate a general Turing machine. In general, these interactions are driven by phonetic rules and are not arbitrary. If certain similar restrictions can be placed on the Sandhi grammar, then the ALSM process may become more tractable. We define a property of Sandhi Grammars called *encapsulaion* that closely resembles properties of context free grammars. With this property, the ALSM Sandhi termination problem becomes decidable.

Let (a, b) be an augmentation rule in the Sandhi grammar. The property of external encapsulation means that no ALSM sequence on a string ending with the letter a can ever produce another string ending with a letter different from a. Similarly, no ALSM sequence on a string beginning with b can produce another string beginning with another letter.

Definition 16 (Externally Encapsulated Sandhi Grammar)**.** *A given Sandhi Grammar $\mathcal{G}$ is said to be Externally Encapsulated if for each rule $(\tau, c) \in \mathcal{R}(a, b)$ such that $\tau \in$ {Pre-Augmentation, Post-Augmentation}:*

- $\forall w \in \Sigma^*, \nexists\, w'c \in \Sigma^*\ (c \neq a)$ *s.t.* $w'c \in \Omega^{\mathcal{G}}_{ALSM}(wa)$
- $\forall w \in \Sigma^*, \nexists\, cw' \in \Sigma^*\ (c \neq b)$ *s.t.* $cw' \in \Omega^{\mathcal{G}}_{ALSM}(bw)$

Remark. A necessary and sufficient condition for a given Sandhi Grammar to be Externally Encapsulated is, if $(\tau, c) \in \mathcal{R}(a, b)$ such that $\tau \in$ {Pre-Augmentation, Post-Augmentation}, then:

- $\forall x \in \Sigma, \forall y \in \Sigma\ (y \neq a), (\text{Post-Substitution}, y) \notin \mathcal{R}(x, a)$
- $\forall x \in \Sigma, \forall y \in \Sigma\ (y \neq b), (\text{Pre-Substitution}, y) \notin \mathcal{R}(b, x)$
- $\forall x \in \Sigma, (\text{Post-Elision}, \epsilon) \notin \mathcal{R}(x, a)$
- $\forall x \in \Sigma, (\text{Pre-Elision}, \epsilon) \notin \mathcal{R}(b, x)$

Let (a, b) be an augmentation rule in the Sandhi grammar. The property of internal encapsulation means that after the application of the augmentation rule corresponding to (a, b), no ALSM sequence on the substring produced between a and b (after augmentation) can replace a or b.

Definition 17 (Internally Encapsulated Sandhi Grammar). *A given Sandhi Grammar $\mathcal{G}$ is said to be Internally Encapsulated if for each rule $(\tau, c) \in \mathcal{R}(a, b)$ such that $\tau \in$ {Pre-Augmentation, Post-Augmentation}:*

- $\forall w \in \Sigma^*, \nexists\, cw' \in \Sigma^*\ (c \neq a)$ *s.t.* $cw' \in \Omega^{\mathcal{G}}_{ALSM}(aw)$
- $\forall w \in \Sigma^*, \nexists\, w'c \in \Sigma^*\ (c \neq b)$ *s.t.* $w'c \in \Omega^{\mathcal{G}}_{ALSM}(wb)$

Remark. A sufficient (but not necessary) condition for a given Sandhi Grammar to be Internally Encapsulated is, if $(\tau, c) \in \mathcal{R}(a, b)$ such that $\tau \in$ {Pre-Augmentation, Post-Augmentation}, then:

- $\forall x \in \Sigma, \forall y \in \Sigma\ (y \neq a)$, (Pre-Substitution, y) $\notin \mathcal{R}(a, x)$
- $\forall x \in \Sigma, \forall y \in \Sigma\ (y \neq b)$, (Post-Substitution, y) $\notin \mathcal{R}(x, b)$
- $\forall x \in \Sigma$, (Pre-Elision, ϵ) $\notin \mathcal{R}(a, x)$
- $\forall x \in \Sigma$, (Post-Elision, ϵ) $\notin \mathcal{R}(x, b)$

Definition 18 (Encapsulated Sandhi Grammar). *A given Sandhi Grammar is said to be Encapsulated if it is both Externally as well as Internally Encapsulated.*

Another important decidability result pertains to the encapsulated Sandhi grammars.

Theorem 7. *The Arbitrary Location Sandhi Merging Termination Problem is decidable for an Encapsulated Sandhi Grammar.*

We now, return to our question about the decidability of Sanskrit Morphophonoligical processes. Unfortunately, the Sanskrit Sandhi Grammar does have some rules due to which the encapsulation property is violated. However, these rules are very small in number and it may be possible to prove the decidability results simular to the case of DLSM termination problem.

Observation. *For the morphophonological Sanskrit Grammar described in Appendix B, only 8 rules defy the property of Encapsulation.*

6 ARBITRARY LOCATION SANDHI SPLITTING (ALSS)

Sandhi Splitting is yet another morphophonological process that is prevalent in the Sanskrit language. It reverses the action of Sandhi on a given input word and splits it into constituent words that can undergo Sandhi Merging to give back the original input word.

Definition 19 (ALSS Termination Problem). *Given an input Sandhi Grammar $\mathcal{G}$ and a word $w \in \Sigma^*$, the ALSM Termination Problem is to determine if $\exists n \in \mathbb{N}$ such that for all $w' \in \Sigma^*$ there exists no ALSM Sequence of length n which produces w from w'.*

Definition 20 (ALSS Production Set). *Given an input Sandhi Grammar $\mathcal{G}$ and a word $w \in \Sigma^*$, the ALSS Production Set $\Omega^{\mathcal{G}}_{ALSS}(w)$ is defined as the set of strings w' such that there exists an ALSM Sequence of finite length which produces w from w'.*

Definition 21 (ALSS Derivation Problem). *Given an input Sandhi Grammar $\mathcal{G}$, a word $w_{in} \in \Sigma^*$ and a word $w_{out} \in \Sigma^*$, the ALSS Derivation Problem is to determine if $w_{out} \in \Omega^{\mathcal{G}}_{ALSS}(w_{in})$.*

Theorem 8. *The Arbitrary Location Sandhi Splitting Termination Problem is undecidable for a general Sandhi Grammar.*

Corollary 8.1. *The Arbitrary Location Sandhi Splitting Derivation Problem is undecidable for a general Sandhi Grammar.*

Generally, we are interested in finding out w_2 from w_1 for ALSS as well as ALSM. The following result shows that one may be obtained from the other.

Theorem 9 (Near Duality). *If given a Sandhi Grammar $\mathcal{G}$, $w' \in \Omega^{\mathcal{G}}_{ALSS}(w)$ then there exists a Sandhi Grammar $\mathcal{G}'$ such that $w' \in \Omega^{\mathcal{G}'}_{ALSM}(w)$, where $\mathcal{G}'$ can be constructed from $\mathcal{G}$. Moreover, $\Omega^{\mathcal{G}'}_{ALSM}(w) \supseteq \Omega^{\mathcal{G}}_{ALSS}(w)$.*

Remark. In general, the above relation is a proper superset. However, when $\mathcal{G}$ does not contain any Pre-Elision or Post-Elision rules, the above two sets turn out to be equal.

7 EXPERIMENTAL RESULTS

According to Theorem 3, DLSM is decidable for the Sanskrit Grammar. This implies that an algorithm to perform Sandhi Merging can be devised. We implement an algorithm that takes as input a Sandhi Grammar $\mathcal{G}$ and two words w_1 and w_2, and produces w after Definite Location Sandhi Merging. This is a restricted form of Sanskrit Sandhi Merging wherein we omit some context and completely ignore syntax and semantics of the Sanskrit rules. The Sandhi Merging Tool created by us outperforms existing tools when evaluated using the Sandhi Merging benchmark of SandhiKosh. The table below compares the results of our DLSM Tool against existing technologies. Our tool shows maximum improvement on the Rule Based corpora as these test the completeness of the underlying Sandhi Grammar. The improvement on the Literature and Bhagavad-Gita corpora is not as significant owing to the fact that existing tools have been optimized on freely available Sanskrit literature which forms the basis of these two corpora.

Corpus	Words	JNU	UoH	INRIA	DLSM Tool
Rule based - Internal	150	21 (14.0%)	36 (24.0%)	79 (52.7%)	137 (91.3%)
Rule based - External	132	38 (28.8%)	57 (29.5%)	67 (50.8%)	122 (92.4%)
Literature	150	53 (35.3%)	130 (86.7%)	128 (85.3%)	142 (94.7%)
Bhagavad-Gita	1430	338 (23.64%)	1045 (73.1%)	1184 (82.1%)	1193 (83.4%)
UoH	9368	3506 (37.4 %)	7480 (79.8%)	7655 (81.7%)	8627 (92.1%)
Astadhyayi	2700	455 (16.9%)	1752 (64.9%)	1762 (65.2%)	2664 (98.7%)

8 CONCLUSION

Morphophonological processes are present in many languages, but hold an important relevance in the case of Sanskrit. We consider an important subset of the Sanskrit Sandhi rules and formulate a Sandhi Grammar based on the immediate interaction of letters (for example across word or morpheme boundaries). We analyze two classes of problems, Sandhi Merging and Sandhi Splitting. For Sandhi Merging we further analyze two variants, Definite Location (DLSM) and Arbitrary Location (ALSM). While for Sandhi Splitting we analyze just the Arbitrary Location variant (ALSS).

We conclude that for a general Sandhi Grammar all three of DLSM, ALSM and ALSS are undecidable. However, the Sanskrit Sandhi Grammar for DLSM (given in Appendix B) turns out to be decidable and we provide an algorithm for it. More generally, we prove DLSM to be decidable if the corresponding STG has no non-negative weight cycles. For ALSM, we prove that excluding either all Elision rules or all Augmentation rules from the Sandhi Grammar makes it decidable. Also we

show that under the special constraint of Encapsulation, ALSM is decidable. The Sanskrit Sandhi Grammar, unfortunately, does not turn out to be Encapsulated with 8 rules defying that property.

Future work in this subject will involve analyzing the decidability of ALSM for the Sanskrit Sandhi Grammar. It shall also include formulating an algorithm to solve ALSS. Finally, more work also needs to be done to discover the precise complexity classes that the constrained variants of ALSM belong to.

ACKNOWLEDGMENTS

We thank Prof. Subodh Kumar (Indian Institute of Technology, Delhi) for providing insightful comments and useful improvements to our paper. We would also like to thank Neelamadhav Gantayat (IBM Research) for his assistance in evaluating the existing Sanskrit Sandhi tools on the standard SandhiKosh corpora.

REFERENCES

Charles H Bennett. 1973. Logical reversibility of computation. *IBM journal of Research and Development* 17, 6 (1973), 525–532.

Surendranath Dasgupta. 1966. *A History of Sanskrit Literature: Classical Period.* Vol. 1. University of Calcutta.

Benjamin W Fortson IV. 2011. *Indo-European language and culture: An introduction.* Vol. 30. John Wiley & Sons.

Huet Gerard. 2018. The Sanskrit Heritage Site. (2018). http://sanskrit.inria.fr/.

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a Sanskrit reader. In *Proceedings, 5th International Symposium on Sanskrit Computational Linguistics. DK Printworld (P) Ltd.* 130–171.

John E Hopcroft and Jeffrey D Ullman. 1969. Formal languages and their relation to automata. (1969).

Girish Nath Jha. 2017. Special Centre for Sanskrit Studies. (2017). http://sanskrit.jnu.ac.in/index.jsp.

Sumitra Mangesh Katre et al. 1989. *Astadhyayi of Panini.* Motilal Banarsidass Publ.

Arthur Berriedale Keith. 1993. *A history of Sanskrit literature.* Motilal Banarsidass Publishe.

Amba Kulkarni. 2017. Saṃsādhanī - A Sanskrit Computational Toolkit. (2017). http://sanskrit.uohyd.ac.in/scl/.

SD 2017. Panini Ashtadhyayi Sutras with Commentaries: Sortable Index. (2017). http://sanskritdocuments.org/learning_tools/ashtadhyayi/.

Bhardwaj Shubham, Neelamadhav Gantayat, Nikhil Chaturvedi, Rahul Garg, and Sumeet Agarwal. 2018. Sandhikosh: A benchmark corpus for evaluating sanskrit sandhi tools. In *11th International Conference on Language Resources and Evaluation.*

JF Staal. 1965. Euclid and Panini. *Philosophy East and West* 15, 2 (1965), 99–116.

Paul Thieme. 1935. *Pānini and the Veda: Studies in the early history of linguistic science in India.* Globe press.

A APPENDIX: RELEVANT PROOFS

Theorem 1. *The Definite Location Sandhi Merging Termination Problem is undecidable for a general Sandhi Grammar.*

Proof. A deterministic Turing Machine can be defined as $M =< Q, \Gamma, b, \Sigma, \delta, q_0, F >$ where:

- Q is a finite, non-empty set of states
- Γ is a finite, non-empty set of tape alphabet symbols
- $b \in \Gamma$ is a blank symbol
- $\Sigma \subseteq \Gamma \backslash \{b\}$ is the set of input symbols
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states
- $\delta : (Q \backslash F) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is a partial function called the transition function, where L is left right and R is right shift. If δ is not defined on current state and tape symbol, M halts.

Given a semi-finite deterministic Turing Machine $M =< Q, \Gamma, b, \Sigma, \delta, q_0, F >$ and an input $w \in \Sigma^*$, we shall construct a Sandhi Grammar, $\mathcal{G} =< \Sigma, \mathcal{R} >$, corresponding input word $w' \in \Sigma^*$ and starting Sandhi Location κ_0. The Sandhi Grammar $\mathcal{G}$ on w' will simulate the action of the Turing Machine M on input w. If M halts on w, the DLSM Sequence of $\mathcal{G}$ on w' starting at Sandhi Location κ_0 will terminate. If M does not halt on w, the DLSM Sequence of $\mathcal{G}$ on w' starting at Sandhi Location κ_0 will not terminate.

To the Sandhi Grammar input alphabet Σ we add the Turing Machine tape alphabet Γ and the Turing Machine state space Q. Along with these we also add the Turing Machine state space Q crossed with L, R to aid the process of head movement. Also needed are two sets of special symbols Θ (to perform state-symbol interaction) and M (to perform head movement). Finally the letter ψ is also added to represent the string boundary.

$$\Sigma = \Gamma \cup Q \cup (Q \times \{L, R\}) \cup \Theta \cup M \cup \psi$$
$$\Theta = \{\theta_k : k \in Q \times \Gamma\}$$
$$M = \{\mu_k : k \in Q \times \Gamma \times \{L, R\}\}$$

Remark. Since Q and Γ are finite, Σ for the Sandhi Grammar will also be finite.

Given the Turing Machine input w and starting state q_0, the corresponding input word w' for the Sandhi Grammar and starting Sandhi Location shall be given as follows:

$$w' = q_0 w \psi$$
$$\kappa_0 = 1$$

Intuition: To simulate any Turing Machine transition $\delta(q, t) = (q', t', m), m \in \{L, R\}$, the state q in the Sandhi Grammar string will interact with the symbol t to the right of it through a series of Interaction Rules following which it will convert itself to q'_m and the symbol to t'. Subsequently, through a series of Movement Rules, the new state q'_m will move itself to the right if $m = R$ or will move itself to the left if $m = L$ and then bring itself to q', thus being ready for the next transition. The interaction and movement will require their corresponding special symbols at intermediary steps. If at any point q reaches the end of the word, it will simply augment the blank symbol b with the help of the special symbol ψ. This will allow the Sandhi Grammar to account for the infiniteness of the Turing Machine tape. Also, since the Turing Machine is deterministic, the constructed Sandhi Grammar will also be deterministic.

For each Turing Machine transition $\delta(q,t) = (q',t',m), m \in \{L,R\}$, we add the following Interaction Rules to $\mathcal{R}$. These rules take the substring $qt \rightarrow q'_m t'$ but do not perform any change to the Sandhi Location.

No.	Input (c_1,c_2)	Output $\mathcal{R}(c_1,c_2)$	Action
IRa	(q,t)	(Post-Substitution, θ_{qt})	$qt \rightarrow q\theta_{qt}$
IRb	(q,θ_{qt})	(Pre-Substitution, q'_m)	$q\theta_{qt} \rightarrow q'_m\theta_{qt}$
IRc	(q'_m,θ_{qt})	(Post-Substitution, t')	$q'_m\theta_{qt} \rightarrow q'_m t'$

To allow for head movement to the right, we add the following Movement Rules to $\mathcal{R}$ for all $q \in Q$ and $t \in \Gamma$. These rules take the substring $q_R t \rightarrow tq$ and increase the Sandhi Location by 1 (through the rule MRc).

No.	Input (c_1,c_2)	Output $\mathcal{R}(c_1,c_2)$	Action
MRa	(q_R,t)	(Post-Substitution, μ_{qtR})	$q_R t \rightarrow q_R\mu_{qtR}$
MRb	(q_R,μ_{qtR})	(Pre-Substitution, t)	$q_R\mu_{qtR} \rightarrow t\mu_{qtR}$
MRc	(t,μ_{qtR})	(Pre-Augmentation, q)	$t\mu_{qtR} \rightarrow tq\mu_{qtR}$
MRd	(q,μ_{qtR})	(Post-Elision, ϵ)	$tq\mu_{qtR} \rightarrow tq$

To allow for head movement to the left, we add the following Movement Rules to $\mathcal{R}$ for all $q \in Q$ and $t,t' \in \Gamma$. These rules take the substring $t'q_L t \rightarrow qt't$ and decrease the Sandhi Location by 1 (through the rule MRf).

No.	Input (c_1,c_2)	Output $\mathcal{R}(c_1,c_2)$	Action
MRe	(q_L,t)	(Post-Augmentation, μ_{qtL})	$q_L t \rightarrow q_L\mu_{qtL}t$
MRf	(q_L,μ_{qtL})	(Pre-Elision, ϵ)	$t'q_L\mu_{qtL} \rightarrow t'\mu_{qtL}$
MRg	(t',μ_{qtL})	(Pre-Substitution, $\mu_{qt'L}$)	$t'\mu_{qtL} \rightarrow \mu_{qt'L}\mu_{qtL}$
MRh	$(\mu_{qt'L},\mu_{qtL})$	(Post-Substitution , t')	$\mu_{qt'L}\mu_{qtL} \rightarrow \mu_{qt'L}t'$
MRi	$(\mu_{qt'L},t')$	(Pre-Substitution , q)	$\mu_{qt'L}t' \rightarrow qt'$

Lastly, to allow for the infiniteness of the Turing Machine, the following Boundary Rule is added to $\mathcal{R}$ for all $q \in Q$. This takes the substring $q\psi \rightarrow qb\psi$ where b is the blank symbol. This rule does not update the Sandhi Location.

No.	Input	Output	Action
BR	(q,ψ)	(Post-Augmentation, b)	$q\psi \rightarrow qb\psi$

Claim: After k ($k \geq 0$) transitions of the Turing Machine M, the Sandhi Grammar $\mathcal{G}$ will be in an equivalent state after k sets of DLSM Steps involving Interaction Rules and corresponding Movement Rules.

That is, the tape content of $M = m_1 m_2 \ldots m_p bbb \ldots$ will be same as the string produced by the DLSM Sequence after omitting the state in the string and symbol ψ, and ignoring trailing blank symbols. Also the state q of M will be equal to the state present at the Sandhi Location κ in the string produced by the DLSM Sequence, and the head position h of M will be equal to the Sandhi Location κ.

Base Case: For $k = 0$, the Turing Machine tape contents are $wbbb\ldots$ and the input string to $\mathcal{G}$ is w after ignoring q_0 and ψ. Furthermore, the initial state q_0 is present in the input string at $\kappa = 1$ which is also equal to the initial head position $h = 1$.

Induction Step: Assuming the above claim holds true for some $k \geq 0$ transitions, we shall show it holds true for $k + 1$ transitions. Let the tape contents of M after k transitions be $m_1 \ldots m_h \ldots m_l$, the state be q and the head position be h. Equivalently, the string produced after a DLSM Sequence involving k sets of Interaction and Movement Rules will be $m_1 \ldots q m_h \ldots m_l \psi$ with $\kappa = h$

Let the Turing Machine perform the transition $\delta(q, m_h) = (q', m'_h, R)$. The Turing Machine tape contents will become $m_1 \ldots m'_h \ldots m_l$, the state will become q' and the head position will become $h+1$. For the Sandhi Grammar, Interaction rules will produce the new string $m_1 \ldots q' m'_h \ldots m_l \psi$ while not changing κ. Following this, Movement Rules will take the string to $m_1 \ldots m'_h q' m_{h+1} \ldots m_l \psi$ and increase κ by 1. Thus, the Sandhi Grammar remains in an equivalent state as the Turing Machine. Note that if after this process q' is followed by ψ in the produced string, we will perform another DLSM Step using the Boundary Rule without breaking equivalence.

Similarly, let the Turing Machine perform the transition $\delta(q, m_h) = (q', m'_h, L)$. The Turing Machine tape contents will become $m_1 \ldots m'_h \ldots m_l$, the state will become q' and the head position will become $h - 1$. For the Sandhi Grammar, Interaction rules will produce the new string $m_1 \ldots q' m'_h \ldots m_l \psi$ while not changing κ. Following this, Movement Rules will take the string to $m_1 \ldots q' m_{h-1} m'_h \ldots m_l \psi$ and increase κ by 1. Thus, the Sandhi Grammar remains in an equivalent state as the Turing Machine.

Having shown that the proposed Sandhi Grammar $\mathcal{G}$ can successfully simulate the action of a given Turing Machine M on an input w, we can conclude that if the Turing Machine terminates, i.e., $\delta(q, m_h)$ is undefined then by construction $\mathcal{R}(q, m_h) = \phi$. On the other hand, if the Turing Machine executes indefinitely, so will the DLSM Sequence of the proposed Sandhi Grammar.

Thus, given a machine that can decide whether a Sandhi Grammar $\mathcal{G}$ will terminate on a given input, we can decide whether a Turing Machine M will halt on an input w. Since the Halting Problem is undecidable, the DLSM Termination Problem is also undecidable. QED. □

Corollary 1.1. *The Definite Location Sandhi Merging Derivation Problem is undecidable for a general Sandhi Grammar.*

Proof. *Intuition:* We will update the Sandhi Grammar $\mathcal{G}$ such that it after the Turing Machine M halts on its input w, $\mathcal{G}$ will perform another set of DLSM Steps that will erase all the existing contents of the string and produce a special string for all starting inputs.

The mapping given in Theorem 1 can be invoked for this proof. Two new symbols ξ_R and ξ_L are added to the Sandhi Grammar input alphabet Σ. Along with that, for all $q \in Q, t \in \Gamma$ such that $\delta(q, t)$ is undefined, the following Clearing Rule is added to $\mathcal{R}$. This rule will modify any state to the special symbol xi_R after the Turing Machine has halted, i.e., $\delta(q, t)$ is undefined.

No.	Input	Output	Action
CRa	(q, t)	(Pre-Substitution, ξ_R)	$q_R t \rightarrow \xi_R t$

Furthermore, for each $t \in \Gamma$, the following Clearing Rules are added to $\mathcal{R}$. These rules will allow xi_R to consume all string contents to the right of it, and will allow xi_L to consume all string contents to the left of it.

No.	Input	Output	Action
CRb	(ξ_R, t)	(Post-Elision, ϵ)	$\xi_R t \rightarrow \xi_R$
CRc	(t, ξ_L)	(Pre-Elision, ϵ)	$t \xi_L \rightarrow \xi_L$

Finally, the following Clearing Rules are also added. These rules will allow xi_R to turn into xi_L once it reaches the string boundary.

No.	Input	Output	Action
CRd	(ξ_R, ψ)	(Post-Substitution, ξ_L)	$\xi_R\psi \rightarrow \xi_R\xi_L$
CRe	(ξ_R, ξ_L)	(Pre-Elision, ϵ)	$\xi_R\xi_L \rightarrow \xi_L$

Thus, once the Turing Machine M halts on its input w, the Sandhi Grammar $\mathcal{G}$ will introduce a new symbol ξ_R to erase all the letters to the right of it, following which the symbol ξ_L will be introduced at the boundary which shall then erase all letters to the left of it, i.e., till it is the only remaining character.

Clearly, if the Turing Machine M halts on its input w, there will be a DLSM Sequence from the starting string $w' = q_{0R}w\psi$ to the single letter string ξ_L. If the Turing Machine does not halt, the string ξ_L will never be reached as no xi_R will ever be introduced.

Given a machine to solve the DLSM Derivation Problem, we can solve the Turing Machine Halting Problem by deciding whether there is a DLSM Sequence from $w' = q_0w\psi$ to ξ_L given $\kappa_0 = 1$. Since the Turing Machine Halting Problem is undecidable, the DLSM Derivation Problem is also undecidable. QED. □

Theorem 2. *If a Sandhi Transition Graph for a given Sandhi Grammar has no non-negative weight cycles, then the DSLM Sequence for every word w given any starting Sandhi Location κ_0 ($\kappa_0 < |w|$) terminates.*

Proof. Given a Sandhi Grammar $\mathcal{G}$ and a corresponding Sandhi Transition Graph (STG) $G =< V, E >$, the STG can mirror any DLSM Sequence of $\mathcal{G}$ on an input w given a starting Sandhi Location κ_0.

Given an initial input $w_0 = c_{01}c_{02}\ldots c_{0n}$ and a Sandhi Location κ_0, the STG traversal starts at vertex $v_0 = (c_{0\kappa_0}, c_{0\kappa_0+1})$ and an edge is traversed for each valid DLSM Step taken.

Claim: There exists a valid sequence of edge traversals such that after i valid DLSM Steps, $v_i = (c_{i\kappa_i}, c_{i\kappa_i+1})$.

The claim trivially holds true for $i = 0$. Assuming the claim holds true after n iterations, $v_n = (c_{n\kappa_n}, c_{n\kappa_n+1})$. If $(\tau, c) \in \mathcal{R}(c_{n\kappa_n}, c_{n\kappa_n+1})$, then:

- $(c_{n+1\kappa_{n+1}}, c_{n+1\kappa_{n+1}+1}) = (c, c_{n\kappa_n+1})$ and $e = (v_n, (c, c_{n\kappa_n+1})) \in E$ if τ = Pre-Augmentation
- $(c_{n+1\kappa_{n+1}}, c_{n+1\kappa_{n+1}+1}) = (c_{n\kappa_n}, c)$ and $e = (v_n, (c_{n\kappa_n}, c)) \in E$ if τ = Post-Augmentation
- $(c_{n+1\kappa_{n+1}}, c_{n+1\kappa_{n+1}+1}) = (c, c_{n\kappa_n+1})$ and $e = (v_n, (c, c_{n\kappa_n+1})) \in E$ if τ = Pre-Substitution
- $(c_{n+1\kappa_{n+1}}, c_{n+1\kappa_{n+1}+1}) = (c_{n\kappa_n}, c)$ and $e = (v_n, (c_{n\kappa_n}, c)) \in E$ if τ = Post-Substitution
- $(c_{n+1\kappa_{n+1}}, c_{n+1\kappa_{n+1}+1}) = (c_{n\kappa_n-1}, c_{n\kappa_n+1})$ and $e = (v_n, (c_{n\kappa_n-1}, c_{n\kappa_n+1})) \in E$ if τ = Pre-Elision
- $(c_{n+1\kappa_{n+1}}, c_{n+1\kappa_{n+1}+1}) = (c_{n\kappa_n}, c_{n\kappa_n+2})$ and $e = (v_n, (c_{n\kappa_n}, c_{n\kappa_n+2})) \in E$ if τ = Pre-Elision

Thus, for $n + 1$ iterations as well there exists a path that takes $v_{n+1} = (c_{n+1\kappa_{n+1}}, c_{n+1\kappa_{n+1}+1})$. Any DLSM Sequence can hence be considered as an equivalent traversal of the STG. If the DLSM Sequence does not terminate, the STG traversal will also not terminate.

Since the STG has a finite number of vertices, an infinite length traversal can occur only with the presence of cycles. If no cycles exist in the STG, there can be no infinite length traversal, and hence there can be no non-terminating DLSM Sequence. Thus, a Sandhi Grammar $\mathcal{G}$ will terminate on any input w given any starting Sandhi Location κ_0 if there are no cycles in the corresponding Sandhi Transition Graph.

Moreover, edge weight of the edge corresponding to a DLSM Step represents the change caused by the DLSM Step to the overall length of the input word. Thus, the net change to the word length

by any DLSM Sequence is equal to the sum of the edge weights of the corresponding edges of each DLSM step of that sequence.

Given an STG cycle of total edge weight l, repeated traversal of that cycle will cause the word length in the corresponding DLSM Sequence to change by l with each repetition. Thus, if the total edge weight is negative, any input word of finite length n will be reduced to a single letter in at max $\lceil n/l \rceil$ traversals of the cycle, leading to termination of the DLSM Sequence.

Thus, a Sandhi Grammar $\mathcal{G}$ will terminate on any input w given any starting Sandhi Location κ_0 if there are no cycles with non-negative weight in the corresponding Sandhi Transition Graph. QED. □

Theorem 3. *For the morphophonological Sanskrit Grammar described in Appendix B, the DLSM Termination Problem is decidable.*

Proof. We run a cycle detection algorithm on the Sandhi Grammar in Appendix B to analyze the decidability of the Sanskrit Sandhi Grammar. Since as per Theorem 2 negative weight cycles do not lead to indefinite DLSM Sequences, we restrict our analysis to non-negative weight cycles. The cycles depending on their size are listed below. All these cycles have a zero weight. For compactness, we merely state the involved rules that are involved in the cycle and not the involved letters.

- **4-cycles:** (65, 67, 63, 67), (66, 67, 63, 67)
- **2-cycles:** (61, 67), (124, 122), (124, 123), (84, 121)
- **1-cycles:** (84), (52), (126), (127)

Claim: Since all the above rules are deterministic, we can decide whether a given input $w = c_1 c_2 \ldots c_n$ and a starting Sandhi Location κ will have a terminating DLSM Sequence on the Sanskrit Sandhi Grammar.

We start the DLSM Sequence and correspondingly traverse the STG starting at $(c_\kappa, c_{\kappa+1})$. If it never reaches any cycle node, then it shall terminate as the number of nodes in the STG is finite. If it does reach a cycle node (involved in a cycle of length k) when the DLSM Sequence produces w', then since all the rules involved in the cycle are deterministic, it will never leave the cycle. Furthermore since the cycle is zero weight, we can create a $G_E^{m+k}S$ as described in Theorem 5 (m = |w'|). All strings produced subsequently by the DLSM process will be closed under this ES Graph $G_E^{m+k}S$. If we can reach a cycle in $G_E^{m+k}S$ starting from w', then the DLSM Sequence for w starting at κ will not terminate. Otherwise, it shall terminate. Thus, For the morphophonological Sanskrit Grammar described in Appendix B, the DLSM Termination Problem is decidable. QED. □

Theorem 4. *The Arbitrary Location Sandhi Merging Termination Problem is undecidable for a general Sandhi Grammar.*

Proof. We invoke the mapping given under Theorem 1 for this proof. By construction, in any intermediary string produced by the Sandhi Grammar $\mathcal{G}$ there will be only one applicable Sandhi Location. Therefore, when ALSM is applied on the same Sandhi Grammar, it will deterministically follow the same sequence as done by DLSM. Hence even ALSM can be simulated to solve the Halting Problem, leading to the undecidability of the Arbitrary Location Sandhi Merging Termination Problem. □

Corollary 4.1. *The Arbitrary Location Sandhi Merging Derivation Problem is undecidable for a general Sandhi Grammar.*

Proof. We invoke the mapping given under Theorem 1 for this proof and its corresponding extension given by Corollary 1.1. As shown in Theorem 4, ALSM will follow the same sequence as done by DLSM. The same idea can also be applied to the extension given by Corollary 1.1. Hence

even ALSM will consume the string after the Turing Machine halts and produce the single letter string ξ_L. Thus, given a machine to solve the ALSM Derivation Problem, we can solve the Turing Machine Halting Problem by deciding whether there is an ALSM Sequence from $w' = q_0 w \psi$ to ξ_L given $\kappa_0 = 1$. Since the Turing Machine Halting Problem is undecidable, the ALSM Derivation Problem is also undecidable. □

Theorem 5. *The Arbitrary Location Sandhi Merging Termination Problem is decidable for a Sandhi Grammar $\mathcal{G} = (\Sigma, \mathcal{R})$ where all rules in $\mathcal{R}$ follow a reduced set of Sandhi Types with no Pre-Augmentation or Post-Augmentation. $\mathcal{R} : \Sigma \times \Sigma \longmapsto \mathcal{T} \times \Sigma$, $\mathcal{T}$ = {Pre-Substitution, Post-Substitution, Pre-Elision, Post-Elision}.*

Proof. Given a Sandhi Grammar $\mathcal{G} =< \Sigma, \mathcal{R} >$ where all rules in $\mathcal{R}$ follow a reduced set of Sandhi Types with no Pre-Augmentation or Post-Augmentation, and an input $w \in \Sigma^*$ ($|w| = n$), we shall construct a directed graph $G^n_{ES} =< V, E >$ called the ES Graph of size n (as it involves only Elision and Substitution) as follows.

The vertices V of the graph shall correspond to all strings of length less than or equal to n, i.e., $V = \Sigma^0 \cup \Sigma^1 \cup \ldots \Sigma^n$. Given $s_1, s_2 \in V$, an edge $e = (s_1, s_2) \in E$ iff there exists a single ALSM Step which can produce s_2 from s_1 given $\mathcal{G}$.

Since the input alphabet Σ is finite, $\Sigma^0 \cup \Sigma^1 \cup \ldots \Sigma^n$ is also finite, and hence V is finite. For every $s \in V$, we traverse the finite Rule-Set $\mathcal{R}$ and populate edges for all valid DLSM Steps. Since $\mathcal{R}$ does not contain any Post-Augmentation or Pre-Augmentation rules, the length of the produced string can never be greater than the length of the input string. Hence, all DLSM Steps will be closed within V.

Given this finite directed graph, we can enumerate all simple cycles using any standard algorithm like Johnson's algorithm. Let C be the set of all $s \in V$ that appear in atleast one simple cycle. Since $C \subseteq V$, C is finite.

Given $w \in \Sigma^n$, if there exists a path from $w \in V$ to any string $s \in C$, then we can conclude the ALSM Sequence for w to be non-terminating since it is possible to construct a non-terminating DLSM Sequence starting from w which can loop forever in a cycle containing s. If no such path exists from $w \in V$ to any string $s \in C$, we can conclude the DLSM Sequence for w to be terminating. If not, then there would be some string that would repeat in the ALSM Sequence that would repeat (since the number of possible strings is finite), and therefore that would correspond to a cycle in G^n_{ES} leading to a contradiction. Thus, the ALSM Termination Problem is decidable for a Sandhi Grammar $\mathcal{G}$ where all rules in $\mathcal{R}$ follow a reduced set of Sandhi Types with no Pre-Augmentation or Post-Augmentation. QED. □

Remark. The same graph created above can also be used to solve the ALSM Derivation Problem given w_{in} and w_{out}.

Theorem 6. *The Arbitrary Location Sandhi Merging Derivation Problem is decidable for a Sandhi Grammar $\mathcal{G} = (\Sigma, \mathcal{R})$ where all rules in $\mathcal{R}$ follow a reduced set of Sandhi Types with no Pre-Elision or Post-Elision. $\mathcal{R} : \Sigma \times \Sigma \longmapsto \mathcal{T} \times \Sigma$, $\mathcal{T}$ = {Pre-Substitution, Post-Substitution, Pre-Augmentation, Post-Augmentation}. Furthermore, such a Grammar can be represented by a Context-Sensitive Grammar.*

Proof. Any Context Sensitive Grammar is defined as a 4-tuple $G =< N, T, P, S >$ where:

- N is a finite set of non-terminal symbols
- T is a finite set of terminal symbols
- $S \in N$ is the start symbol
- P is the finite set of production rules of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$, where $A \in N$, $\gamma \in (N \cup T)^*$ and $\alpha, \beta \in (N \cup T)^+$

Given a Sandhi Grammar $\mathcal{G} = (\Sigma, \mathcal{R})$ where all rules in $\mathcal{R}$ follow a reduced set of Sandhi Types with no Pre-Elision or Post-Elision, a corresponding Context Sensitive Grammar $G =< N, T, P, S >$ can be constructed such that $L(G) = L = \{w_2,w_1^R$: Given $\mathcal{G}$ there exists a finite ALSM Sequence from w_1 that produces $w_2\}$.

Given $\Sigma = \{c_1, c_2, \ldots, c_n\}$, $\Sigma_N = \{C_1, C_2, \ldots C_n\}$ where for each $c_i \in \Sigma$ there is a corresponding $C_i \in \Sigma_N$. The Context Sensitive Grammar $G =< N, T, P, S >$ can be constructed as follows:

$$N = \Sigma_N \cup \{S\}$$
$$T = \Sigma \cup \{,\}$$

The following production rules are added to P:

$$S \rightarrow , \tag{1}$$
$$S \rightarrow C_i S c_i \qquad \forall i \in \{1 \ldots n\} \tag{2}$$
$$C_i \rightarrow c_i \qquad \forall i \in \{1 \ldots n\} \tag{3}$$

If $(\tau, c_k) \in \mathcal{R}(c_i, c_j)$, we add the following production rules to P:

$$C_i C_j \rightarrow C_k C_j \qquad \text{if } \tau = \text{Pre-Substitution} \tag{4}$$
$$C_i C_j \rightarrow C_i C_k \qquad \text{if } \tau = \text{Post-Substitution} \tag{5}$$
$$C_i C_j \rightarrow C_i C_k C_j \qquad \text{if } \tau = \text{Pre-Augmentation} \tag{6}$$
$$C_i C_j \rightarrow C_i C_k C_j \qquad \text{if } \tau = \text{Post-Augmentation} \tag{7}$$

In simple terms, the above production rules produce the string w_1 in reverse to the right of comma, and a string for w_1 made of corresponding non-terminals to the left of the comma. The non-terminals then interact as if performing an ALSM Sequence and give rise to new strings. At any point in the sequence, the generation can be terminated and the non-terminals can be taken to their corresponding terminals. Since a Context Sensitive Grammar is non-contracting, we cannot encode the Elision rules of a Sandhi Grammar in the above paradigm.

To show that $L(G) \supseteq L$, if $s = w_2,w_1^R \in L$ then there exists an ALSM Sequence of finite length from w_1 to w_2 given $\mathcal{G}$. Let $w_1 = c_{i_1} c_{i_2} \ldots c_{i_m}, \forall j\ i_j \in \{1 \ldots n\}$. $S \Rightarrow C_{i_1} C_{i_2} \ldots C_{i_m}, c_{i_m} \ldots c_{i_2} c_{i_1}$ is a valid derivation achieved by m applications of production rule (2) followed by the application of production rule (1). Let $W_1 = C_{i_1} C_{i_2} \ldots C_{i_m}$. Hence, $S \Rightarrow W_1,w_1^R$.

For every ALSM Step in $\mathcal{G}$ that takes some input w to w', there exists a corresponding production rule in G that can take W to W'. Since an ALSM Sequence is a sequence of ALSM Steps, is there exists an ALSM Sequence from w to some w'', then there will exist a sequence of production rules that can take W to the corresponding W''. Hence, given that an ALSM Sequence from w_1 to w_2 exists, a sequence of production rules that take W_1 to W_2 also exists. $S \Rightarrow W_2,w_1^R$. Following this, after m applications of production rule (3) $S \Rightarrow w_2,w_1^R$. Therefore, if $s \in L$ then $s \in L(G)$. Hence proved, $L(G) \supseteq L$.

To show that $L(G) \subseteq L$, we first see that any valid sequence of production rules that produces a string $s \in L(G)$ can be reordered to an equivalent sequence of a specific ordering that produces the same string.

Claim: All production sequences can be reordered such that all applications of rule (2) happen at the start, followed by rule (1), followed by all applications of rules (4)-(7), ending with all applications of rule (3).

Since rule (3) does not affect the non-terminal S, and once it is applied for some non-terminal C_i, there can be no further action involving C_i through rules (4)-(7) subsequently in the sequence. Thus, all applications of rule (3) can be pushed to the end of the production sequence. Clearly, no sequence can have an application of rule (1) before any application of rule (2) since it terminates

the non-terminal S. Now, since no action on a non-terminal C_i is possible before its addition in any production sequence, we can simply move the application of any rule (2) above the application of rules (4)-(7) above it. The same holds true for rule (1) since it does not affect the action of rules (4)-(7). Following this procedure, the production sequence is as described in the claim and still produces the same string as before.

At the end of the application of rules (2) and (1), the string will be $S \Rightarrow W_1,w_1^R$ for some $w_1 \in \Sigma^*$ where W_1 is the corresponding string of non-terminals. Suppose the subsequent application of rules (4)-(7) take the string to $S \Rightarrow W_2,w_1^R$. Since rules (4)-(7) mimic the action of an ALSM Sequence, if there is a sequence of production rules (4)-(7) that take W_1 to W_2, then there will also exist a corresponding ALSM Sequence that takes w_1 to w_2. Therefore, after the applications of rule (3), $S \Rightarrow w_2,w_1^R$ where there is an ALSM Sequence from w_1 to w_2. Thus, $s = w_2,w_1^R \in L$. Hence proved, $L(G) \subseteq L$.

The above shows that the Context Sensitive Grammar G as described above accepts the language $L = \{w_2,w_1^R$: Given $\mathcal{G}$ there exists a finite ALSM Sequence from w_1 that produces $w_2\}$. Given w_{in} and w_{out} solving the ALSM Derivation Problem is equivalent to deciding whether $w_{out},w_{in}^R \in L(G)$, which is a decidable problem for Context Sensitive Grammars [Hopcroft and Ullman 1969]. Hence, the Arbitrary Location Sandhi Merging Derivation Problem is decidable for a Sandhi Grammar $\mathcal{G} = (\Sigma, \mathcal{R})$ where all rules in $\mathcal{R}$ follow a reduced set of Sandhi Types with no Pre-Elision or Post-Elision. QED. □

Theorem 7. *The Arbitrary Location Sandhi Merging Termination Problem is decidable for an Encapsulated Sandhi Grammar.*

Proof. Given a word $w \in \Sigma^*$, we present an algorithm that can decide whether the ALSM Sequence for w will terminate on an Encapsulated Sandhi Grammar $\mathcal{G}$.

Firstly, we construct an Augmentation Graph $G_{Aug} =< V, E >$ for $\mathcal{G}$. The vertices of G_{Aug} will be $V \subseteq \Sigma \times \Sigma$ such that if $(\{\text{Pre-Augmentation/Post-Augmentation}\}, c) \in \mathcal{R}(a, b)$ for some c then $(a, b) \in V$. The edges E of G_{Aug} will denote 'reachability' of one Augmentation from another, i.e., if in the expansion of a given Augmentation rule involving (a_1, b_1) we encounter another Augmentation rule involving (a_2, b_2), then there exists $e = ((a_1, b_1), (a_2, b_2)) \in E$.

To populate E, we start with a pair of letters (a, b) such that we have $(\{\text{Pre-Augmentation/Post-Augmentation }\}, c) \in \mathcal{R}(a, b)$ for some $c \in \Sigma$. Now, we apply the rule on string ab and take it to the string acb. As done in Theorem 5, we construct an ES Graph G_{ES}^3 of size 3 using the Elision and Substitution rules in $\mathcal{G}$. Since the given Sandhi Grammar is Encapsulated, all strings that are reachable from abc will have the form axb for some $x \in \Sigma$ or ab. If there exists a cycle in G_{ES}^3 reachable from ab, then in G_{Aug} we add an edge from (a, b) to itself, i.e, $((a, b), (a, b)) \in E$. Moreover, if $(\{\text{Pre-Augmentation/Post-Augmentation}\}, y) \in \mathcal{R}(a, x)$ for some $y \in \Sigma$, we create an edge $((a, b), (a, x)) \in E$. Similarly, if $(\{\text{Pre-Augmentation/Post-Augmentation}\}, z) \in \mathcal{R}(x, b)$ for some $z \in \Sigma$, we create an edge $((a, b), (x, b)) \in E$. We repeat the above process for every Augmentation rule in $\mathcal{G}$.

Now, given $w \in \Sigma^*$, we identify the Sandhi Locations in w where a Pre-Augmentation or a Post-Augmentation rule can be applied, and split $w = w_1w_2 \dots w_k$, such that for all i, if last letter of w_i is c_{i1} and first letter of w_{i+1} is c_{i2}, then $(\{\text{Pre-Augmentation/Post-Augmentation}\}, c') \in \mathcal{R}(c_{i1}, c_{i2})$ for some $c' \in \Sigma$. Since $\mathcal{G}$ is Internally Encapsulated, $\mathcal{R}(c_{i1}, c_{i2})$ can have no Elision or Substitution rule that destroys c_{i1} or c_{i2}.

Given $w = w_1w_2 \dots w_k$, let $m = max_i\{|w_i|\}$. We now create the ES Graph G_{ES}^m. Since $\mathcal{G}$ is Externally Encapsulated, all strings in G_{ES}^m reachable from w_i will end with c_{i1} (last letter of w_i) and all strings reachable from w_{i+1} will start with c_{i2} (first letter of w_{i+1}). This is because any letters

involved in an Augmentation rule cannot be destroyed or consumed in an Encapsulated Sandhi Grammar.

If for any w_i there exists a cycle in G^m_{ES} reachable from w_i, we decide that the ALSM Sequence for w is non-terminating. This is because we can always produce an ALSM Sequence from w_i, and hence from w that can reach that cycle of strings in G^m_{ES} and loop forever.

If no cycle is reachable for any w_i in G^m_{ES}, let T_i denote the set of strings reachable from w_i that have no outgoing edge in G^m_{ES}. T_i will not be empty since no cycle is reachable from w_i and G^m_{ES} is finite. Given a string $w'_i \in T_i$, it may be possible that more Augmentation rules may be valid in w'_i. Thus, w'_i may be split on the Sandhi Locations in where a Pre-Augmentation or a Post-Augmentation rule can be applied leading to $w'_i = w'_{i1}w'_{i2}\dots w'_{ik_i}$. Note that since there is no outgoing edge from w'_i in G^m_{ES}, there is no applicable Substitution or Elision rule for w'_i or for any w'_{ij}. Also note that for every string w''_i that exists on the path between w_i and w'_i in G^m_{ES}, if a certain Augmentation rule will be applicable for w''_i, then that Augmentation rule will also be applicable for w'_i (since the Sandhi Grammar is Encapsulated and no letters involved in an Augmentation rule can be destroyed or consumed once produced).

Thus, selecting one terminal from each T_i, $w' = w'_{11}w'_{12}\dots w'_{1k_1}w'_{21}w'_{22}\dots w'_{2k_2}\dots w'_{k1}w'_{k2}\dots w'_{kk_k}$ such that there exists an ALSM Sequence from w to w'. Consider every boundary pair (a, b) where a is the last letter of w'_{ij} and b is the first letter of $w'_{i(j+1)}$ if $j \neq k_i$ or b is the first letter of $w'_{(i+1)1}$ if $j = k_i$. If for any combination of terminals, any boundary pair (a, b) can reach a cycle in the Augmentation Graph G_{Aug}, then the ALSM Sequence for w will not terminate. This is because we will always be able to construct an ALSM Sequence from ab, and hence from w, which will get stuck in a cycle and loop forever.

If for no combination of terminals, no boundary pair (a, b) can reach a cycle in the Augmentation Graph G_{Aug}, then the ALSM Sequence for w will terminate. If not, then w can either loop forever in a cycle of repeating strings or grow indefinitely. If it gets stuck in a cycle of repeating strings, it implies that at least one w_i could reach a cycle in the G^m_{ES} graph since no two w_i can interact with each other across a boundary. Hence, leading to a contradiction. Else, if w grows forever, it means it is stuck in a cycle where the same Augmentation rules get repeated (since the number of Augmentation rules is finite). But, since no boundary pair in any terminal string reachable from w has any cycle reachable from it in the Augmentation Graph G_{Aug}, the above is a contraction. Hence, the ALSM Termination Problem is decidable for an Encapsulated Sandhi Grammar. □

Theorem 8. *The Arbitrary Location Sandhi Splitting Termination Problem is undecidable for a general Sandhi Grammar.*

Proof. This proof goes on the same lines as Theorem 1. Owing to lack of space, we shall omit the details and present the basic idea. Instead of creating the mapping from a general Turing Machine, we shall construct the mapping given an equivalent Reversible Turing Machine (RTM) [Bennett 1973].

Since Sandhi Splitting can function on arbitrary locations, all Augmentation and Elision rules given under Interaction and Movement Rules can be replaced by equivalent Substitution rules (since we do not need to explicitly update κ). This makes it easier to reverse their action. Given any Interaction or Movement Rule (Pre-Substitution, c) $\in \mathcal{R}(a, b)$ we shall replace it with (Pre-Substitution, a) $\in \mathcal{R}(c, b)$. Given any Interaction or Movement Rule (Post-Substitution, c) $\in \mathcal{R}(a, b)$ we shall replace it with (Pre-Substitution, b) $\in \mathcal{R}(a, c)$.

The only non-substitution rule that shall remain will be the Boundary Rule, the action of which can be reversed by replacing it with the rule (Pre-Elision, ϵ) $\in \mathcal{R}(b, \psi)$, where b is the blank symbol.

Clearly, if the earlier Sandhi Grammar had an ALSM path from w to w', the new mapping will have an ALSM path from w' to w, implying an ALSS path from w to w' (by definition). Hence, this new Sandhi Grammar can simulate a given Reversible Turing Machine. Since the Halting Problem is undecidable, the ALSS Termination Problem will also be undecidable. QED. □

Corollary 8.1. *The Arbitrary Location Sandhi Splitting Derivation Problem is undecidable for a general Sandhi Grammar.*

Proof. Given an input Sandhi Grammar $\mathcal{G}$, a word $w_{in} \in \Sigma^*$ and a word $w_{out} \in \Sigma^*$, the ALSS Derivation Problem is to determine if $w_{out} \in \Omega^{\mathcal{G}}_{ALSS}(w_{in})$. From Corollary 4.1, we know that the ALSM Derivation Problem is undecidable for a general Sandhi Grammar.

If the ALSS Derivation Problem is decidable, then given $w_1, w_2 \in \Sigma^*$ we can decide $w_2 \in \Omega^{\mathcal{G}}_{ALSM}(w_1)$ if $w_1 \in \Omega^{\mathcal{G}}_{ALSS}(w_2)$ (by definition $w_1 \in \Omega^{\mathcal{G}}_{ALSS}(w_2)$ if there exists a finite ALSM Sequence from w_1 that produces w_2). Thus, the ALSS Derivation Problem is undecidable for a general Sandhi Grammar. QED. □

Theorem 9 (Near Duality). *If given a Sandhi Grammar $\mathcal{G}$, $w' \in \Omega^{\mathcal{G}}_{ALSS}(w)$ then there exists a Sandhi Grammar $\mathcal{G}'$ such that $w' \in \Omega^{\mathcal{G}'}_{ALSM}(w)$, where $\mathcal{G}'$ can be constructed from $\mathcal{G}$. Moreover, $\Omega^{\mathcal{G}'}_{ALSM}(w) \supseteq \Omega^{\mathcal{G}}_{ALSS}(w)$.*

Proof. The above claim is equivalent to saying that if given a Sandhi Grammar $\mathcal{G}$ there exists a finite ALSM Sequence from w to w', then it is possible to construct a Sandhi Grammar $\mathcal{G}'$ given which there exists a finite ALSM Sequence from w' to w.

$\mathcal{G}' =< \Sigma', \mathcal{R}' >$ can be constructed as follows. $\Sigma' = \Sigma$. If $(\tau, c) \in \mathcal{R}(c_1, c_2)$, then:

- (Pre-Elision, ϵ) $\in \mathcal{R}'(c, c_2)$ if τ = Pre-Augmentation
- (Post-Elision, ϵ) $\in \mathcal{R}'(c_1, c)$ if τ = Post-Augmentation
- (Pre-Substitution, c_1) $\in \mathcal{R}'(c, c_2)$ if τ = Pre-Substitution
- (Post-Substitution, c_2) $\in \mathcal{R}'(c_1, c)$ if τ = Post-Substitution
- $\forall c' \in \Sigma$, (Pre-Augmentation, c_1) $\in \mathcal{R}'(c', c_2)$ if τ = Pre-Elision
- $\forall c' \in \Sigma$, (Post-Augmentation, c_2) $\in \mathcal{R}'(c_1, c')$ if τ = Post-Elision

Claim: For every ALSM Step s given $\mathcal{G}$ that produces w_2 from w_1 by action on some arbitrary Applicable Sandhi Location κ, there exists a corresponding ALSM Step s^{-1} given $\mathcal{G}'$ that produces w_1 from w_2.

Given $w = c_1 c_2 \dots c_n$ $(\tau, c) \in \mathcal{R}(c_\kappa, c_{\kappa+1})$, the above can be shown as follows:

$$c1 \dots c_\kappa c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}(c_\kappa, c_{\kappa+1})]{(Pre-Aug, c)} c1 \dots c_\kappa c c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}'(c, c_{\kappa+1})]{(Pre-Elision, \epsilon)} c1 \dots c_\kappa c_{\kappa+1} \dots c_n$$

$$c1 \dots c_\kappa c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}(c_\kappa, c_{\kappa+1})]{(Post-Aug, c)} c1 \dots c_\kappa c c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}'(c_\kappa, c)]{(Post-Elision, \epsilon)} c1 \dots c_\kappa c_{\kappa+1} \dots c_n$$

$$c1 \dots c_\kappa c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}(c_\kappa, c_{\kappa+1})]{(Pre-Subst, c)} c1 \dots c c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}'(c, c_{\kappa+1})]{(Pre-Subst, c_\kappa)} c1 \dots c_\kappa c_{\kappa+1} \dots c_n$$

$$c1 \dots c_\kappa c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}(c_\kappa, c_{\kappa+1})]{(Post-Subst, c)} c1 \dots c_\kappa c \dots c_n \xrightarrow[\mathcal{R}'(c_\kappa, c)]{(Post-Subst, c_{\kappa+1})} c1 \dots c_\kappa c_{\kappa+1} \dots c_n$$

$$c1 \dots c_{\kappa-1} c_\kappa c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}(c_\kappa, c_{\kappa+1})]{(Pre-Elision, \epsilon)} c1 \dots c_{\kappa-1} c_{\kappa+1} \dots c_n \xrightarrow[\mathcal{R}'(c_{\kappa-1}, c_{\kappa+1})]{(Pre-Aug, c_\kappa)} c1 \dots c_{\kappa-1} c_\kappa c_{\kappa+1} \dots c_n$$

$$c1 \dots c_\kappa c_{\kappa+1} c_{\kappa+2} \dots c_n \xrightarrow[\mathcal{R}(c_\kappa, c_{\kappa+1})]{(Post-Elision, \epsilon)} c1 \dots c_\kappa c_{\kappa+2} \dots c_n \xrightarrow[\mathcal{R}'(c_\kappa, c_{\kappa+2})]{(Post-Aug, c_\kappa)} c1 \dots c_\kappa c_{\kappa+1} c_{\kappa+2} \dots c_n$$

Thus, for every ALSM Step s in $\mathcal{G}$, there exists s^{-1} in $\mathcal{G}'$ that reverses the effect of s. Let w be obtained from w' given $\mathcal{G}$ through a finite ALSM Sequence of length n. Let w_i be the word produced after i ALSM Steps of that sequence. Thus $w' = w_0$ and $w = w_n$.

Since w_n is produced from w_{n-1} using a single ALSM Step given $\mathcal{G}$, we can produce w_{n-1} from w_n using a single ALSM Step given $\mathcal{G}'$. Similarly w_{n-2} can be produced from w_{n-1}, w_{n-3} from w_{n-2} and so on till w_0 from w_1. Thus, we have been able to show that an ALSM Sequence of finite length can produce $w_0 = w'$ from $w_n = w$ given $\mathcal{G}'$.

Therefore, if $w' \in \Omega^{\mathcal{G}}_{ALSS}(w)$, then $w' \in \Omega^{\mathcal{G}'}_{ALSM}(w)$. It follows trivially that $\Omega^{\mathcal{G}'}_{ALSM}(w) \supseteq \Omega^{\mathcal{G}}_{ALSS}(w)$. QED. □

B APPENDIX: SANSKRIT MORPHOPHONOLOGICAL GRAMMAR

SK#	First Letter	Second Letter	Type	Modification
47	i I	a A u U f F x e E o O	Pre-Substitution	y y
47	u U	a A i I f F x e E o O	Pre-Substitution	v v
47	f F	a A i I u U x e E o O	Pre-Substitution	r r
47	x	a A i I u U f F e E o O	Pre-Substitution	l
52	k K g G c C j J w W q Q t T d D p P b B	g G j J q Q d D b B	Pre-Substitution	g g g g j j j j q q q q d d d d b b b b
61	e E o O	a A i I u U f F x e E o O	Pre-Substitution	ay Ay av Av
63	o O	y	Pre-Substitution	av Av
65	i	y	Pre-Substitution	ay
66	i	y	Pre-Substitution	ay
67	y v	a A i I u U f F x e E o O g G N j J Y q Q R d D n b B m y r l v h	Pre-Elision	
69	a A	i I u U	Combination	e e o o
70	a A	f F x	Combination	ar ar al
72	a A	e E o O	Combination	E E O O
73	a A	e	Combination	E
73	a A	U	Combination	O
73	a	U	Combination	O
73	a	I	Combination	E
73	a	e	Combination	E
73	a	f	Combination	Ar
73	a	f	Combination	Ar
74	a A	f	Combination	Ar
76	r	k K c C w W t T p P S z s	Pre-Substitution	H
78	a A	e o	Combination	e o
78	a	a	Combination	a
78	a	I	Combination	I
78	i	I	Combination	I
80	a A	o	Combination	o
81	a	i	Combination	i
82	a	i	Combination	e
84	k K g G c C j J w W q Q t T d D p P b B s z S h	a A i I u U f F x e E o O k K g G N c C j J Y w W q Q R t T d D n p P b B m y r l v s z S h	Pre-Substitution	g g g g j j j j q q q q d d d d b b b b d w j g
85	a A	a A	Combination	A A
85	i I	i I	Combination	I I
85	u U	u U	Combination	U U
85	F	f F	Combination	F F
85	f	f x	Combination	F F
85	f	f x	Combination	f x
86	e o	a	Combination	e o

88	o	a A i I u U f F x e E o O	Pre-Substitution	ava
89	o	i	Pre-Substitution	ava
91	i	a A u U	Concatenation	
91	u	a A u U	Concatenation	
91	I	a A u U	Pre-Substitution	i
91	U	a A u U	Pre-Substitution	u
92	a i u	f	Concatenation	
92	A I U	f	Pre-Substitution	a i u
101	I U	a A i I u U f F x e E o O	Concatenation	
111	t T d D n s	c C j J Y S	Pre-Substitution	c C j J Y S
111	c C j J Y	t T d D n s	Post-Substitution	c C j J Y S
111	S	s	Post-Substitution	S
113	t T d D n s	w W q Q R	Pre-Substitution	w W q Q R z
113	w W q Q R z	t T d D n s	Post-Substitution	w W q Q R z
113	s	z	Pre-Substitution	z
114	w W q Q R	t T d D n s	Concatenation	
114	w W q Q R	t T d D n s	Post-Substitution	w W q Q R z
115	k K g G c C j J w W q Q t T d D p P b B	N Y R n m	Pre-Substitution	N N N N Y Y Y Y R R R R n n n n m m m m
116	t T d D n	l	Pre-Substitution	l l l l l~
118	d	s	Post-Elision	
119	g G j J q Q d D b B	h	Post-Substitution	G
120	k K c C j J t T p P	S	Post-Substitution	C
121	g G j J q Q d D b B	k K c C w W t T p P S z s	Pre-Substitution	k k c c w w t t p p
122	m	k K g G N c C j J Y w W q Q R t T d D n p P b B m y r l v s z S h	Pre-Substitution	M
123	n m	k K g G c C j J w W q Q t T d D p P b B s z S h	Pre-Substitution	M M
124	M	k K g G N c C j J Y w W q Q R t T d D n p P b B m	Pre-Substitution	N N N N N Y Y Y Y Y R R R R R n n n n n m m m m m m
125	M	k K g G N c C j J Y w W q Q R t T d D n p P b B m	Pre-Substitution	N N N N N Y Y Y Y Y R R R R R n n n n n m m m m m m
126	m	r	Pre-Substitution	m
127	m	h	Pre-Substitution	m
127	m	h	Pre-Substitution	y
127	m	h	Pre-Substitution	v
127	m	h	Pre-Substitution	l
129	n	h	Pre-Substitution	m
130	N R	S z s	Pre-Augmentation	k w

131	q	s	Post-Augmentation	D
132	n	s	Post-Augmentation	D
133	n	S	Pre-Augmentation	t
134	N R n	a A i I u U f F x e E o O	Post-Augmentation	N R n
138	H	k K c C w W t T p P S z s	Pre-Substitution	s
138	H	k K c C w W t T p P S z s	Pre-Substitution	s
139	m	k K c C w W t T p P	Pre-Substitution	r
140	n	c C t T w W	Pre-Substitution	r
141	n	p	Pre-Substitution	r
143	n	k	Pre-Substitution	r
144	H	k p	Pre-Substitution	s
144	H	k p	Pre-Substitution	z
144	H	k p	Pre-Substitution	s
146	a i u f x	C	Pre-Augmentation	t
147	A	C	Pre-Augmentation	t
148	A I U F	C	Pre-Augmentation	t