From Paninian Sandhi to Finite State Calculus

Malcolm D. Hyman

From Pāṇinian Sandhi to Finite State Calculus

Malcolm D. Hyman[*]
Max Planck Institute for the History of Science, Berlin

Abstract

The most authoritative description of the morphophonemic rules that apply at word boundaries (external sandhi) in Sanskrit is by the great grammarian Pāṇini (fl. 5th c. B.C.E.). These rules are stated formally in Pāṇini's grammar, the *Aṣṭādhyāyī* 'group of eight chapters'. The present paper summarizes Pāṇini's handling of sandhi, his notational conventions, and formal properties of his theory. An XML vocabulary for expressing Pāṇini's morphophonemic rules is then introduced, in which his rules for sandhi have been expressed. Although Pāṇini's notation potentially exceeds a finite state grammar in power, individual rules do not rewrite their own output, and thus they may be automatically translated into a rule cascade from which a finite state transducer can be compiled.

1 Sandhi in Sanskrit

Sanskrit possesses a set of morphophonemic rules (both obligatory and optional) that apply at morpheme and word boundaries (the latter are also termed *pada boundaries*). The former are called *internal sandhi* (< *saṃdhi* 'putting together'); the latter, *external sandhi.* This paper only considers external sandhi. Sandhi rules involve processes such as assimilation and vowel coalescence. Some examples of external sandhi are: *na asti* > *nāsti* 'is not', *tat ca* > *tac ca* 'and this', *etat hi* > *etad dhi* 'for this', *devas api* > *devo 'pi* 'also a god'.[1]

2 Sandhi in Pāṇini's grammar

Pāṇini's *Aṣṭādhyāyī* is a complete grammar of Sanskrit, covering phonology, morphology, syntax, semantics, and even pragmatics. It contains about 4000 rules (termed *sūtra,* literally 'thread'), divided between eight chapters (termed *adhyāya*). Conciseness (*lāghava*) is a fundamental principle in Pāṇini's formulation of carefully interrelated rules (Smith, 1992). Rules are either *operational* (i. e. they specify a particular linguistic operation, or *kārya*) or *interpretive* (i. e. they define the scope of operational rules).[2] Rules may be either obligatory or optional.

A brief review of some well-known aspects of Pāṇini's grammar is in order. The operational rules relevant to sandhi specify that a substituend (*sthānin*) is replaced by a substituens (*ādeśa*) in a given context (Cardona, 1965b, 308). Rules are written using metalinguistic case conventions, so that the substituend is marked as genitive, the substituens as nominative, the left context as ablative (*tasmāt*), and the right context as locative (*tasmin*). For instance:

8.4.62	*jhayo*	*ho*	*'nyatarasyām*
	jhaY-ABL	*h*-GEN	optionally

This rule specifies that (optionally) a homogenous sound replaces *h* when preceded by a sound termed *jhaY* — i. e. an oral stop (Sharma, 2003, 783–784). Pāṇini uses abbreviatory labels (termed *pratyāhāra*) to describe phonological classes. These labels are interpreted in the context of an ancillary text of the *Aṣṭādhyāyī*, the *Śivasūtras,* which enumerate a catalog of sounds (*varṇasamāmnāya*) in fourteen classes (Cardona, 1969, 6):

1. a i u Ṇ
2. r̥ l̥ K
3. e o Ṅ
4. ai au C
5. h y v r Ṭ
6. l Ṇ
7. ñ m ṅ ṇ n M
8. jh bh Ñ
9. gh ḍh dh Ṣ
10. j b g ḍ d Ś
11. kh ph ch ṭh th c ṭ t V
12. k p Y
13. ś ṣ s R
14. h L

[*]This work has been supported by NSF grant IIS-0535207. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the National Science Foundation. The paper has benefited from comments by Peter M. Scharf and by four anonymous referees.

[1]The symbol ⟨'⟩ (*avagraha*) does not represent a phoneme but is an orthographic convention to indicate the prodelision of an initial *a*-.

[2]The traditional classification of rules is more fine-grained and comprises *saṃjñā* (technical terms), *paribhāṣā* (interpretive rules), *vidhi* (operational rules), *niyama* (restriction rules), *pratiṣedha* (negation rules), *atideśa* (extension rules), *vibhāṣā* (optional rules), *nipātana* (ad hoc rules), *adhikāra* (heading rules) (Sharma, 1987, 89).

The final items (indicated here by capital letters) are markers termed *it* 'indicatory sound' and are not considered to belong to the class. A *pratyāhāra* formed from a sound and an *it* denotes all sounds in the sequence beginning with the specified sound and ending with the last sound before the *it.* Thus *jhaY* denotes the class of all sounds from *jh* through *p* (before the *it Y*): *jh, bh, gh, ḍh, dh, j, b, g, ḍ, d, kh, ph, ch, ṭh, th, c ṭ, t, k, p* (i. e. all oral stops).[3] Sūtra 8.4.62, as printed above, is by itself both elliptic and uninterpretable. Ellipses in sūtras are completed by supplying elements that occur in earlier sūtras; the device by which omitted elements can be inferred from preceding sūtras is termed *anuvr̥tti* 'recurrence' (Sharma, 1987, 60). It will be noticed that no substituens is specified in 8.4.62; the substituens *savarnaḥ* 'homogenous sound-NOM' (Cardona, 1965a) is supplied by *anuvr̥tti* from sūtra 8.4.58 *anusvārasya yayi parasavarṇaḥ.* Still, the device of *anuvr̥tti* is insufficient to specify the exact sound that must be introduced as a substituens. It is here that interpretive rules play a role. Sūtra 8.4.62 must be interpreted in the light of the *paribhāṣā* 'interpretive rule' 1.1.50 *sthāne 'ntaratamaḥ*, which specifies that a substituens must be maximally similar (sc. in articulatory place and manner) to the substituend (Sharma, 1987, 126). Thus the substituens in 8.4.62 will always be aspirated, since *h* (the substituend) is aspirated: e. g. *vāg hasati* (< *vāk hasati* 'a voice laughs' by 8.2.39) > *vāgghasati.*[4]

A second example further illustrates the principles already discussed:

8.4.63	*śaś*	*cho*	*'ṭi*
	ś-GEN	*ch*-NOM	*aṬ*-LOC

Here *jhayaḥ* '*jhaY*-ABL' and *anyatarasyām* 'optionally' are supplied by *anuvr̥tti* from the preceding sūtra (8.4.62). The rule specifies that (optionally) *ś* is replaced by *ch* when it is preceded by an oral stop (*jhaY*) and followed by a vowel or semivowel (*aṬ*).

Rules specific to external sandhi are found in the third quarter (*pāda*) of the eighth *adhyāya.* A number of rules are common to both internal and external sandhi, and rules relevant for external sandhi are also found in the first *pāda* of the sixth *adhyāya* and the fourth *pāda* of the eighth *adhyāya.*

3 An XML encoding for Pāṇinian rules

An encoding based on Extensible Markup Language (XML) has been chosen for expressing Pāṇinian rules in machine-readable form. The XML vocabulary contains an element `<rule>`, with required attributes `source` (the substituend) and `target` (the substituens) and optional attributes `lcontext` (the left context) and `rcontext` (the right context). The values of `source`, `lcontext`, and `rcontext` are specified as Perl-compatible regular expressions (PCREs) (Wall et al., 2000).[5] Sanskrit sounds are indicated in an encoding known as SLP1 (Sanskrit Library Phonological 1) (Scharf and Hyman, 2007). An encoding such as Unicode is not used, since Unicode represents *written characters* rather than *speech sounds* (Unicode Consortium, 2006). The SLP1 encoding facilitates linguistic processing by representing each Sanskrit sound with a single symbol (see fig. 1).[6] The use of Unicode would be undesirable here, since (1) there would not be a one-to-one correspondence between character and sound, and (2) Sanskrit is commonly written in a number of different scripts (Devanāgarī, Tamil, Romanization, etc.).

The following is the XML representation of sūtra 8.3.23 *mo 'nusvāraḥ,* which specifies that a *pada*-final *m* is replaced by the nasal sound *anusvāra* (*ṃ*) when followed by a consonant (Sharma, 2003, 628):

```
<rule source="m" target="M"
      rcontext="[@(wb)][@(hal)]"
      ref="A.8.3.23"/>
```

This rule employs a syntactic extension used for *macros.* The expression `@(`*`name`*`)` is replaced by the value of a defined macro *`name`*. Macros are defined with an XML element `<macro>` and may be defined recursively. Macro expansion is performed immediately after parsing the XML file, before any further processing. Here the `rcontext` attribute references two macros: `@(wb)` is expanded to characters that indicate a word boundary, and `@(hal)` is expanded to the characters representing the sounds of the *pratyāhāra haL* (i. e. all consonants). Macros are a syntactic convenience that allows rules to be easier to read (and closer to Pāṇini's original formulation); one might equally spell out in full all characters representing sounds in a phonological class. The square brackets belong to the PCRE syntax and indicate that any character contained between them should be matched; that is, `[abc]` matches an `a`, `b`, or `c`.

In the `target` two additional syntactic facilities allow for rules to be expressed in a way that is close to Pāṇini's formulation. These facilities are termed *mappings* and *functions.* The following rule illustrates a mapping:

```
<rule source="h"
      target="%(voicedaspirate($1))"
      lcontext="([@(Jay)])[@(wb)]"
      optional="yes"
      ref="A.8.4.62"/>
```

[3]The vowel *a* added after a consonant makes the *pratyāhāra* pronounceable.

[4]Note that Sanskrit *h* represents a voiced glottal fricative [ɦ].

[5]So-called "regular expressions" in programming languages such as Perl include extended features such as pattern memory that exceed the power of regular languages (see §4); for example, it is possible to write a regular expression that matches the $\alpha\alpha$ language, that is, the language of all reduplicated strings. Thus PCREs are *not,* in the formal sense, regular expressions at all.

[6]Figure 1 is a simplified overview of SLP1. It does not show symbols for accents, certain nasalized sounds, or sounds peculiar to the Vedic language.

अ a **a**	आ ā **A**	इ i **i**	ई ī **I**	उ u **u**	ऊ ū **U**
ऋ r̥ **f**	ॠ r̥̄ **F**	ऌ l̥ **x**	ॡ l̥̄ **X**	*	
ए e **e**	ऐ ai **E**	ओ o **o**	औ au **O**		
क् k **k**	ख् kh **K**	ग् g **g**	घ् gh **G**	ङ् ṅ **N**	
च् c **c**	छ् ch **C**	ज् j **j**	झ् jh **J**	ञ् ñ **Y**	
ट् ṭ **w**	ठ् ṭh **W**	ड् ḍ **q**	ढ् ḍh **Q**	ण् ṇ **R**	
त् t **t**	थ् th **T**	द् d **d**	ध् dh **D**	न् n **n**	
प् p **p**	फ् ph **P**	ब् b **b**	भ् bh **B**	म् m **m**	
य् y **y**	र् r **r**	ल् l **l**	व् v **v**		
श् ś **S**	ष् ṣ **z**	स् s **s**	ह् h **h**		

* anusvāra = **M**; visarga = **H**

Figure 1: A partial overview of the SLP1 encoding

This sūtra has been discussed earlier. The left context matches a *jhaY* followed by a word boundary. The parentheses (part of PCRE syntax) specify that the contents (the *jhaY*) be stored in pattern memory. Strings stored in pattern memory are available for subsequent reference: the pattern matched by the first parenthesized group may be recalled with `$1`; the second, with `$2`, etc. (only nine pattern memory variables are available). The pattern memory variable `$1` is referenced in the `target` attribute. The rule specifies that an *h*, when preceded by a *pada*-final *jhaY*, is replaced by the result of performing the `voicedaspirate` mapping on the matched *jhaY.* The mapping syntax takes the form `%(`*`name`*`(`*`input`*`))`, where *`name`* is the name of a mapping, and *`input`* is the input symbol for the mapping. A mapping is defined with a `<mapping>` element, which has as children one or more `<map>` elements. The mapping `voicedaspirate` is defined thus:

```
<mapping name="voicedaspirate">
  <map from="@(jaS)" to="@(Jaz)"/>
</mapping>
```

This mapping translates the voiced oral stops denoted by the *pratyāhāra jaŚ* (*j, b, g, ḍ, d*) to the equivalent aspirated voiced oral stops denoted by the *pratyāhāra jhaṢ* (*jh, bh, gh, ḍh, dh*). In the case that the input symbol to a mapping is not contained in `from`, the mapping is equivalent to the identity function.

Sūtra 6.1.87 *ād guṇaḥ* illustrates the use of a function:

```
<rule source="[@(a)][@(wb)]([@(ik)])"
      target="!(gunate($1))"
      ref="A.6.1.87"/>
```

The `source` matches either short *a* or long *ā* (by the definition of the macro `a`), a word boundary, and then the *pratyāhāra iK* (a simple vowel other than *a* or *ā*). The macro `ik` is defined:

```
<macro name="ik"
       value="@(i)@(u)@(f)@(x)"
       ref="A.1.1.71"/>
```

and depends on the macro definitions:

```
<macro name="i" value="iI"
       ref="A.1.1.69"/>
<macro name="u" value="uU"
       ref="A.1.1.69"/>
<macro name="f" value="fF"
       ref="A.1.1.69"/>
<macro name="x" value="xX"
       ref="A.1.1.69"/>
```

The *iK* is stored in pattern memory, and the substituend (*a-varṇa*) is replaced by the output of calling the function `gunate` on the stored *iK.* A function is defined with an element `<function>`, which has as children one or more `<rule>` elements. These are context-free rules that typically make use of mappings in the `target`. Thus `gunate` is defined:

```
<function name="gunate">
  <rule source="[@(a)@(i)@(u)]"
        target="%(guna($1))"/>
  <rule source="[@(f)@(x)]"
        target="%(guna($1))
                %(semivowel($1))"/>
</function>
```

Two mappings are invoked here:

```
<mapping name="guna"
         ref="A.1.1.2">
  <map from="@(a)" to="a"/>
  <map from="@(i)" to="e"/>
  <map from="@(u)" to="o"/>
  <map from="@(f)" to="a"/>
  <map from="@(x)" to="a"/>
</mapping>

<mapping name="semivowel"
         ref="A.6.1.77">
  <map from="@(i)" to="y"/>
  <map from="@(u)" to="v"/>
  <map from="@(f)" to="r"/>
  <map from="@(x)" to="l"/>
</mapping>
```

The mapping `guna` maps simple vowels such as *a-varṇa* (which includes short *a* and long *ā*) to their *guṇa* equivalent. The term *guṇa* is defined by the technical rule (*saṃjñā*) (Sharma, 1987, 102) 1.1.2 *adeṅ guṇaḥ* '*a* and *eṄ* [are] *guṇa*' (the *pratyāhāra eṄ* = {*e, o*}). The mapping `semivowel` maps those vowels that possess homorganic semivowels to the corresponding semivowels. The function `gunate`, if the input is *a-*, *i-*, or *u-varṇa,* outputs the corresponding *guṇa* vowel; if the input is *r̥-* or *l̥-varṇa,* it outputs the corresponding *guṇa* vowel (in this case, *a*) concatenated with the homorganic semivowel (either *r* or *l*). The domain of `gunate` is {*a, ā, i, ī, u, ū, r̥, r̥̄, l̥, l̥̄*} and its range is {*a, e, o, ar, al*}.

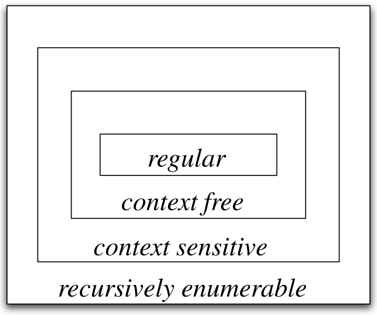

Figure 2: The Chomsky hierarchy of languages (after Prusinkiewicz and Lindenmayer (1990, 3))

4 Regular languages and regular relations

First, it is necessary to define a *regular language.* Let Σ denote a finite alphabet and Σ^ϵ denote $\Sigma \cup \{\epsilon\}$ (where ϵ is the empty string). $\{e\}$ is a regular language where $e \in \Sigma^\epsilon$, and the empty language $\emptyset$ is a regular language. Given that L_1, L_2, and L are regular languages, additional regular languages may be defined by three operations (under which they are closed): concatenation ($L_1 \cdot L_2 = \{xy|x \in L_1, y \in L_2\}$), union ($L_1 \cup L_2$), and Kleene closure ($L^* = \cup_{i=0}^{\infty} L^i$) (Kaplan and Kay, 1994, 338).

Regular relations are defined in the same fashion. An n-relation is a set whose members are ordered n-tuples (Beesley and Karttunen, 2003, 20). Then $\{e\}$ is a regular n-relation where $e \in \Sigma^\epsilon \times \ldots \times \Sigma^\epsilon$ ($\emptyset$ is also a regular n-relation). Given that R_1, R_2, and R are regular n-relations, additional regular n-relations may be defined by three operations (under which they are closed): n-way concatenation ($R_1 \cdot R_2 = \{xy|x \in R_1, y \in R_2\}$), union ($R_1 \cup R_2$), and n-way Kleene closure ($R^* = \cup_{i=0}^{\infty} R^i$).

5 Finite state automata and regular grammars

A *finite state automaton* is a mathematical model that corresponds to a regular language or regular relation (Beesley and Karttunen, 2003, 44). A simple finite state automaton corresponds to a regular language and is a quintuple $\langle S, \Sigma, \delta, s_0, F\rangle$, where S is a finite set of states, Σ the alphabet of the automaton, δ is a transition function that maps $S \times \Sigma^\epsilon$ to 2^S, $s_0 \in S$ is a single initial state, and $F \subseteq S$ is a set of final states (Aho et al., 1988, 114). A finite state automaton that corresponds to a regular relation is termed a *finite state transducer* (FST) and can be defined as a quintuple $\langle S, \Sigma \times \ldots \times \Sigma, \delta, s_0, F\rangle$, with δ being a transition function that maps $S \times \Sigma^\epsilon \times \ldots \times \Sigma^\epsilon$ to 2^S (Kaplan and Kay, 1994, 340).

A regular (or finite) grammar describes a regular language and is equivalent to a finite state automaton. In a regular grammar, all production rules have a single non-terminal on the left-hand side, and either a single terminal or a combination of a single non-terminal and a single terminal on the right-hand side. That is, all rules are of the form $A \rightarrow a$, $A \rightarrow aB$ (for a right regular grammar), or $A \rightarrow Ba$ (for a left regular grammar), where A and B are single non-terminals, and a is a single terminal (or ϵ). A regular grammar is the least powerful type of grammar in the Chomsky hierarchy (see fig. 2) (Chomsky, 1956). A context free grammar describes a context free language, a context sensitive grammar describes a context sensitive language, and an unrestricted grammar describes a recursively enumerable language. The hierarchy is characterized by proper inclusion, so that every regular language is context free, every context free language is context sensitive, etc. (but not every context free language is regular, etc.). A regular grammar cannot describe a context free language such as $\{a^n b^n | 1 \leq n\}$, which consists of the strings $\{ab, aabb, aaabbb, aaaabbbb \ldots\}$ (Kaplan and Kay, 1994, 346).

Since Pāṇinian external sandhi can be modeled using finite state grammar, it is highly desirable to provide a finite state implementation, which is computationally efficient. Finite state machines are closed under composition, and thus sandhi operations may be composed with other finite state operations to yield a single network.

6 From rewrite rules to regular grammars

A string rewriting system is a system that can transform a given string by means of rewrite rules. A rewrite rule specifies that a substring $x_1 \ldots x_n$ is replaced by a substring $y_1 \ldots y_m$: $x_1 \ldots x_n \rightarrow y_1 \ldots y_m$, where $x_i, y_i \in \Sigma$ (and Σ is a finite alphabet). Rewrite rules used in phonology have the general form

$$\phi \rightarrow \psi / \lambda \underline{\quad\quad} \rho$$

Such a rule specifies that ϕ is replaced by ψ when it is preceded by λ and followed by ρ. Traditionally, the phonological component of a natural-language grammar has been conceived of as an ordered series of rewrite rules (sometimes termed a *cascade*) $W_1, \ldots, W_n$ (Chomsky and Halle, 1968, 20). Most phonological rules, however, can be expressed as regular relations (Beesley and Karttunen, 2003, 33). But if a rewrite rule is allowed to rewrite a substring introduced by an earlier application of the same rule, the rewrite rule exceeds the power of regular relations (Kaplan and Kay, 1994, 346). In practice, few such rules are posited by phonologists. Although certain marginal morphophonological phenomena (such as arbitrary center embedding and unlimited reduplication) exceed finite state power, the vast majority of (morpho)phonological processes may be expressed by regular relations (Beesley and Karttunen, 2003, 419).

Since phonological rewrite rules can normally (with the provisos discussed above) be reexpressed as regular relations, they may be modeled as finite state transducers (FSTs). FSTs are closed under composition; thus if T_1 and T_2 are FSTs, application of the composed transducer $T_1 \circ T_2$ to a string S is equivalent to applying T_1 to S and applying T_2 to the output of T_1:

$$T_1 \circ T_2(S) = T_2(T_1(S))$$

So if a cascade of rewrite rules $W_1, \ldots, W_n$ can be expressed as a series of FSTs $T_1, \ldots, T_n$, there is a single FST T that is a composition $T_1 \circ \ldots \circ T_n$ and is equivalent to the cascade $W_1, \ldots, W_n$ (Kaplan and Kay, 1994, 364). $G = T_1 \circ \ldots \circ T_n$ constitutes a regular grammar. Efficient algorithms are known for compiling a cascade of rewrite rules into an FST (Mohri and Sproat, 1996).

7 An FST for Pāṇinian sandhi

The XML formalism for expressing Pāṇinian rules in §3 contains a number of devices; it is not immediately evident how rules employing these devices might be compiled into an FST. A rule compiler, however, is described here that translates Pāṇinian rules expressed in the XML formalism into rewrite rules that can be automatically compiled into an FST using standard algorithms.

It is useful to begin with an instance of Pāṇinian sandhi derivation of the string *devo 'pi* < *devas api* 'also a god'.

FORM	SŪTRA
`devas api`	
`deva$ api`	8.2.66
`deva#u api`	6.1.113
`dev!(gunate(u)) api`	6.1.87
`devo 'pi`	6.1.109

Sūtra 8.2.66 replaces a *pada*-final *s* with *rU* (here symbolized by `$`). Sūtra 6.1.113 replaces *pada*-final *rU* with *u* preceded by a boundary marker (`#`) when the next *pada* begins with *a*. Sūtra 6.1.87 has been discussed above. Sūtra 6.1.109 replaces *pada*-initial *a* with *avagraha* (represented by `'` in SLP1) when the previous *pada* ends in the *pratyāhāra eṄ* (i. e. *e* or *o*).[7]

Although the PCREs in the XML format exceed the power of regular relations, and the implementation of mappings and functions is not obvious in a regular grammar, the rule compiler mentioned above is able to produce a cascade of rewrite rules that may be efficiently compiled into an FST. A consequence of the rule compilation strategy is that a single Pāṇinian rule may be represented as several rewrite rules. Where a rule makes use of pattern memory, which can contain

[7]Although *avagraha* is not a phoneme, it may be conceived of linguistically as a "trace" left after the deletion of *a*-. For this reason, it is represented in the SLP1 phonological coding.

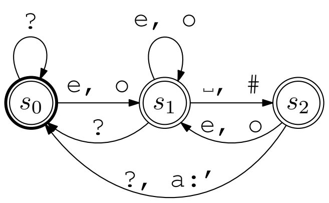

Figure 3: An FST for sūtra 6.1.109

n possible values, the rule is expanded into n rewrite rules. Mappings and functions are automatically applied where they occur in ψ (the substituens).

The following cascade represents the four sūtras exemplified above:

```
s → $ / ____ (␣ | #)
$ → #u / a ____ (␣ |#)a
(a|A)(␣ | #)(a|A) → a
(a|A)(␣ | #)(i|I) → e
(a|A)(␣ | #)(u|U) → o
(a|A)(␣ | #)(f|F) → ar
(a|A)(␣ | #)(x|X) → al
a → ' / (e|o)(␣ | #) ____
```

Here the notation $(x|y|z)$ expresses alternation; the rule matches either x, or y, or z. The symbol ␣ represents a space character; the symbol # represents a boundary that has been inserted by a rule.

These rules may be efficiently compiled into FSTs. An FST encoding the final rule is shown in fig. 3. In this transducer, s_0 is the initial state and doubly-circled states are the members of F, the set of final states. The graphical representation of the FST has been simplified, so that symbols that are accepted on transition from s_i to s_j share an arc between s_i and s_j (in this case, the symbols are separated by commas). The notation $x : y$ indicates the symbols on the upper and lower tapes of the transducer, respectively. If the transducer reads input from the upper tape and writes output to the lower tape, $x : y$ indicates that if x is read on the upper tape, y is written on the lower tape. If the symbols on the upper and lower tapes are the same, a shorthand notation is used; thus x is equivalent to $x : x$. The special symbol ? indicates that *any* symbol is matched on either the upper or lower tape.

Since it is possible to translate each rule in the above cascade into an FST, and FSTs are closed under composition, it is possible to compose a single FST that implements the portion of Pāṇini's sandhi represented by the rules in the cascade above. A compiler developed by the author will be capable of compiling the entirety of Pāṇini's sandhi rules into a single FST. The FST is then converted to Java code using a toolkit written by the author. Compilation of the generated source code yields binary code that may be run portably using any Java Virtual Machine (JVM) (Lindholm and Yellin, 1999). Alternatively, for improved performance, the Java code may be compiled into native machine code using the GNU Compiler for Java (gcj).

8 Implications

While one of the guiding principles of Pāṇini's grammar is conciseness (*lāghava*), a computational implementation poses other demands, such as tractability and efficiency. Pāṇini's rules are formulated in terms of classes based on distinctive features and must be construed with the aid of 'interpretive' (*paribhāṣā*) rules, technical terms (*saṃjñā*), and other theoretical apparatus.

However desirable the mathematical properties of a regular grammar may be, a grammar stated in such terms is at odds with Pāṇini's principles. Rather, it hearkens back to the *vikāra* system employed by earlier linguistic thinkers, in which individual segments are the target of specific rules (Cardona, 1965b, 311).

By way of contrast, Pāṇini states a rule economically in terms of the sound classes enumerated in the *Śivasūtras*. Thus 8.4.41 *ṣṭunā ṣṭuḥ* specifies the retroflexion of an *s* or dental stop either (1) before a *pada*-final *-ṣ* or (2) *pada*-finally before a *ṭU* (i. e. a retroflex stop).

With a little less economy, we can represent Pāṇini's rule by two rules in XML:

```
<rule source="[s@(tu)]"
      target="%(retroflex($1))"
      lcontext="z[@(wb)]"
      ref="A.8.4.41"/>

<rule source="[s@(tu)]"
      target="%(retroflex($1)"
      rcontext="[@(wb)][@(wu)]"
      ref="A.8.4.41"/>
```

The *pratyāhāra tU* stands for the dental stop series {*t, th, d, dh, n*}. We apply the retroflex mapping:

```
<mapping name="retroflex">
  <map from="s" to "z"/>
  <map from="@(tu)" to "@(wu)"/>
</mapping>
```

The effect is to change a single phonological feature across an entire class; sounds with a dental place of articulation (*dantya*) are replaced by sounds with a retroflex articulation (*mūrdhanya*). In specifying such a replacement, Pāṇini makes use of the principle of *sāvarṇya* 'homogeneity of sounds'. So the substituend chosen is that closest to the original—with respect to voice (*ghoṣavat / aghoṣa*), aspiration (*mahāprāṇa / alpaprāṇa*), and nasality (*sānunāsika / niranunāsika*). Thus *t* $\longrightarrow$ *ṭ, th* $\longrightarrow$ *ṭh, d* $\longrightarrow$ *ḍ,* and so on; yet Pāṇini does not need explicitly (and repetitively) to specify the exact segments substituted. In this way, the *Aṣṭādhyāyī* avoids the bias ("segmentalism") that places the linear segment at the center of phonological theory—a bias from which contemporary linguistics is beginning to distance itself (Aronoff, 1992).

The finite state approach discussed in this paper, however, is limited to describing the relations between strings (sequences of segments). As far as the computational model is concerned, individual symbols are atomic, and no class relations between the symbols exist. The goal in this study has been to develop an intermediate representational structure, based on XML, that can faithfully encode some of the linguistically significant aspects of a portion of Pāṇini's grammar (the rules involved in external sandhi) and at the same can be automatically translated into an efficient computational implementation.

9 Appendix: core rules for external sandhi

The following is a list of modeled *vidhi* rules (8.3.4: *adhikāra*, 8.4.44: *pratiṣedha*) for external sandhi. No account is taken here of *pragṛhya* rules that specify certain sounds as exempt from sandhi (since these rules refer to morphosyntactic categories). Various optional rules are not listed.

6.1.73	*che ca*
6.1.74	*āṅmāṅośca*
6.1.132	*etattadoḥ sulopo 'koranañsamāse hali*
8.2.66	*sasajuṣo ruḥ*
8.2.68	*ahan*
6.1.113	*ato roraplutādaplute*
6.1.114	*haśi ca*
6.1.101	*akaḥ savarṇe dīrghaḥ*
6.1.88	*vṛddhireci*
6.1.87	*ādguṇaḥ*
6.1.77	*iko yaṇaci*
6.1.109	*eṅaḥ padāntādati*
6.1.78	*eco 'yavāyāvaḥ*
8.2.39	*jhalāṃ jaśo 'nte*
8.3.4	*anunāsikātparo 'nusvāraḥ*
8.3.7	*naśchavyapraśān*
8.3.14	*ro ri*
8.3.17	*bhobhagoaghoapūrvasya yo 'śi*
8.3.15	*kharavasānayorvisarjanīyaḥ*
8.3.19	*lopaḥ śākalyasya*
8.3.20	*oto gārgyasya*
8.3.23	*mo 'nusvāraḥ*
8.3.31	*śi tuk*
8.3.32	*ṅamo hrasvādaci ṅamuṇnityam*
8.3.34	*visarjanīyasya saḥ*
8.3.35	*śarpare visarjanīyaḥ*
8.3.40	*namaspurasorgatyoḥ*
8.4.40	*stoḥ ścunā ścuḥ*
8.4.44	*śāt*
8.4.41	*ṣṭunā ṣṭuḥ*
8.4.45	*yaro 'nunāsike 'nunāsiko vā*
8.4.53	*jhalāṃ jaśjhaśi*
8.4.55	*khari ca*
8.4.60	*torli*
8.4.62	*jhayo ho 'nyatarasyām*
8.4.63	*śaścho 'ṭi*
8.4.65	*jharo jhari savarṇe*

References

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1988. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Reading, MA.

Mark Aronoff. 1992. Segmentalism in linguistics: The alphabetic basis of phonological theory. In Pamela Downing, Susan D. Lima, and Michael Noonan, editors, *The Linguistics of Literacy*, volume 21 of *Typological Studies in Language*, pages 71–82. John Benjamins, Amsterdam.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI, Stanford, CA.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1996. *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India, New Delhi.

George Cardona. 1965a. On Pāṇini's morphophonemic principles. *Language*, 41(2):225–237.

George Cardona. 1965b. On translating and formalizing Pāṇinian rules. *Journal of the Oriental Institute, Baroda*, 14:306–314.

George Cardona. 1969. Studies in Indian grammarians: I. The method of description reflected in the Śivasūtras. *Transactions of the American Philosophical Society*, 59(1):3–48.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. MIT Press, Cambridge, MA.

Noam Chomsky. 1956. Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3):113–124.

Ronald M. Kaplan and Matin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):332–378.

Tim Lindholm and Frank Yellin. 1999. *The Java^{TM} Virtual Machine Specification*. Addison-Wesley, Reading, MA, 2d edition.

Mehryar Mohri and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 231–238, Santa Cruz, CA. ACL.

Prezemyslaw Prusinkiewicz and Aristid Lindenmayer. 1990. *The Algorithmic Beauty of Plants*. Springer, New York.

Peter M. Scharf and Malcolm D. Hyman. 2007. Linguistic issues in encoding Sanskrit. Unpublished manuscript, Brown University.

Rama Nath Sharma. 1987. *The Aṣṭādhyāyī of Pāṇini, Vol. 1: Introduction to the Aṣṭādhyāyī as a Grammatical Device*. Munshiram Manoharlal, New Delhi.

Rama Nath Sharma. 2003. *The Aṣṭādhyāyī of Pāṇini, Vol. 6: English Translation of Adhyāyas Seven and Eight with Sanskrit Text, Transliteration, Word-Boundary, Anuvṛtti, Vṛtti, Explanatory Notes, Derivational History of Examples, and Indices*. Munshiram Manoharlal, New Delhi.

Henry Smith. 1992. Brevity in Pāṇini. *Journal of Indian Philosophy*, 20:133–147.

Unicode Consortium. 2006. *The Unicode Standard, Version 5.0*. Addison-Wesley, Boston.

Larry Wall, Tom Christiansen, and John Orwant. 2000. *Programming Perl*. O'Reilly, Sebastapol, CA, 3d edition.