On Pāṇini and the Generative Capacity of Contextualized Replacement Systems

Gerald PENN[1] *Paul KIPARSKY*[2]
(1) University of Toronto
(2) Stanford University
gpenn@cs.toronto.edu, kiparsky@csli.stanford.edu

Abstract

This paper re-examines the widely held belief that the formalism underlying the rule system propounded by the ancient Indian grammarian, Pāṇini (ca. 450–350 BCE), either anticipates or converges upon the same expressive power found in finite state control systems or the context-free languages that are used in programming language theory and computational linguistics. While there is indeed a striking but cosmetic resemblance to the contextualized rewriting systems used by modern morphologists and phonologists, a subtle difference in how rules are prevented from applying cyclically leads to a massive difference in generative capacity. The formalism behind Pāṇinian grammar, in fact, generates string languages not even contained within any of the multiple-component tree-adjoining languages, MCTAL(k), for any k. There is ample evidence, nevertheless, that Pāṇini's grammar itself judiciously avoided the potential pitfalls of this unconstrained formalism to articulate a large-coverage, but seemingly very tractable grammar of the Sanskrit language.

Keywords: generative capacity, grammar formalisms, Pāṇini, morphophonological rewriting systems.

1 Background: Formal Language Complexity

Assuming that every language can be characterised as the set of all and only those strings that are grammatical in that language, Chomsky (1959) defined a chain of language classes (sets of languages, thus sets of sets of strings) now called the Chomsky Hierarchy, each class of which is defined by a kind of grammar that can characterise every language in that class. The chain, as Chomsky (1959) defined it, is:

$$RL \subset CFL \subset CSL \subset UL$$

where RL are the regular languages, CFL are the context-free languages, CSL are the context-sensitive languages, and UL are the unrestricted languages.

Note that we are discussing *languages* and not *grammars*. A language is regular (resp. context-free, context-sensitive, unrestricted) if and only if there exists a regular (resp. context-free, context-sensitive, unrestricted) grammar that generates it. Even regular languages have presentations as context-free or context-sensitive grammars, for example, because regular languages are also context-free languages and context-sensitive languages. Even if the context-sensitive rules in a grammar have non-empty contexts, this does not guarantee, *pace* Staal (1965), that the language defined by the grammar is in fact properly context-sensitive. There may be a different presentation of the same language that is a regular grammar. In this case, the language would in fact be a regular language.

Membership in a language class has practical consequences because it determines the worst-case running time of an algorithm that receives a grammar G and string w as input and determines whether w belongs to the language characterised by G. It also arguably has psycholinguistic consequences in that the precise position(s) of human languages relative to these classes has not yet been determined. There are proofs that at least one human language is not syntactically context-free (Swiss-German; Huybregts, 1984, Shieber, 1985) and that at least one human language is not morphologically context-free (Bambara; Culy, 1985).

Normally, within the field of formal language theory, we investigate abstract, nonsensical languages that have simple, precise definitions. The well-known language $\{a^n b^n \mid n \geq 0\}$, for example, is comprised of the strings *ab*, *aabb*, *aaabbb*, etc. This language belongs to CFL, but not to RL. Note that string membership can still be determined in time linear in the length of an input string for this fixed language. But a general CFL membership algorithm would take roughly cubic time.

2 Two Essential Questions of Formal Language Complexity for Pāṇinian Grammar

There are two principal question schemes that we can distinguish with respect to the study of Pāṇinian grammar as a computational device:

A: Given the specific, fixed grammar that Pāṇini articulated in the Aṣṭādhyāyī, which formal language class(es) does it belong to?

B: Given the grammar formalism that Pāṇini used for this grammar, what kind of grammars can we write in general? That is to say, where does the class of Pāṇinian languages fit within the Chomsky hierarchy?

The answer to (A) has been argued by Hyman (2007) to be RL. (B) is much more difficult to answer conclusively because Pāṇini did not define this class formally. But we do have a very thorough example, as well as the benefit of several traditional commentators who speculated as to the unstated conventions that Pāṇini must have assumed in order for the Aṣṭādhyāyī to make correct predictions about Sanskrit grammar.

There have been 2 replies to (B) thus far:

(i) the Pāṇinian languages are the CFL. This answer is widely assumed within computer science circles, probably as a result of a claim by Ingerman (1967) that Pāṇini had anticipated the invention of Backus-Naur form, a means of specifying context-free grammars. Even to the trifling extent that the Aṣṭādhyāyī looks anything like BNF, it would be a strident oversimplification to claim that the two are equivalent in a formal-language-theoretic sense.

(ii) the Pāṇinian languages are either RL or UL. There is an unmistakable similarity between the form of many of the rules in the Aṣṭādhyāyī and the more recent occidental tradition of formulating rules in both phonology and morphology as instances of:

$$\phi \longrightarrow \psi \;/\; \lambda \_\_ \rho,$$

which signifies that an instance of ϕ rewrites to an instance of ψ when preceded by an instance of λ and followed by an instance of ρ. ϕ, ψ, λ and ρ are either strings or (regular) sets of strings. Johnson (1970) proved (without reference to Pāṇini) that systems of these rules generate UL in general, but that with one restriction, which modern morphophonologists seem willing to follow, they only generate RL. That restriction is acyclicity.

The purpose of this paper is to set the record straight on (B). Section 3 discusses the acyclicity restriction in more detail. It turns out that Pāṇini observes a related condition that we shall call Nīlakaṇṭhadīkṣitar's condition. Section 4 proves that the two restrictions, while related, are not equivalent, as can be seen prominently when the redex lengths in individual rules are greater than 1. Section 5 shows that Pāṇini does in fact use redexes of length greater than 1 in his grammar. Section 6 then shows that the Pāṇinian formalism recognizes all of the counting languages, placing it well above context-free in its generative capacity.

3 Acyclicity

In every derivation in one of these contextualized replacement systems, it is possible to relate the rule application instances of the derivation such that $r_1 < r_2$ iff the input redex of r_2 contains at least 1 of the output characters of r_1. The transitive closure of this relation is a partial order, which can be decomposed into totally ordered chains of rule application instances, each successive member of which rewrites some of the output of the previous member. The aforementioned restriction is that there must exist a natural number k such that no rule has more than k application instances in any chain in any derivation.

Often it is assumed by linguists that k must be 1. This is not actually necessary. But often this restriction is paraphrased as: 'no rule may rewrite its own output.' This paraphrase is simply inaccurate; it does not capture how these rules are allowed to interact, even when $k = 1$.

It appears that this "acyclicity" condition does in fact hold of derivations induced by the Aṣṭādhyāyī. But it does so contingently: there are explicit meta-rules in the Aṣṭādhyāyī that seem to have been placed there to establish this restriction (Joshi and Kiparsky, 1979). This means that, unless otherwise stated, Pāṇini's contextualized rules can apply in cycles (and possibly to their own output).

These pre-emptory meta-rules are only used where a prohibition on re-using the same context (λ and ρ) would not already have accomplished the same. Such a prohibition is nowhere explicitly stated in the Aṣṭādhyāyī. Thus the prohibition on re-using contexts does seem to be part of the underlying formalism, and it has been acknowledged as such by at least one traditional commentator (Nīlakaṇṭhadīkṣitar: lakṣye lakṣaṇam sakṛd eva pravartate).

Note that for both Johnson (1970) and Pāṇini, the rules:

$$\phi \longrightarrow \psi \;/\; \lambda \_\_ \rho$$

and:

$$\lambda\phi\rho \longrightarrow \lambda\psi\rho$$

are potentially very different in their effects, as a result.

4 Is Nīlakaṇṭhadīkṣitar's condition equivalent to Acyclicity?

No. Acyclicity implies the former, which can only be violated if the lengths of input and output are equal in some chain of rule applications.

Nīlakaṇṭhadīkṣitar's condition only prevents cyclicity when, in a chain of rule applications, (1) the input and output lengths are equal and (2) that length is 1 or the left and right contexts conspire to prevent partial overlaps between the output of a rule application and the input of a later application of the same rule. To consider a chain of length 2, for example, the rules:

$$aa \longrightarrow bb \;/\; c \_\_ d$$
$$bb \longrightarrow aa \;/\; c \_\_ d$$

by themselves constitute a system that will not accept any string with the substrings *caad* or *cbbd*, and passes through any other input unchanged, if neither of these restrictions is in force. With either acyclicity or Nīlakaṇṭhadīkṣitar's condition (it does not matter which), all input is passed through unchanged, even when it contains caad or cbbd. In this rule system, however:

$$aa \longrightarrow bb \;/\; b \_\_ a$$
$$b \longrightarrow a \;/\; b \_\_ a$$
$$b \longrightarrow a \;/\; \_\_ bb$$

Nīlakaṇṭhadīkṣitar's condition would not be sufficient to prevent cyclic rule applications. On the string *baaaa*, for example, acyclicity produces *baaaa* and *abaaa*, whereas Nīlakaṇṭhadīkṣitar's condition allows *baaaa*, *aabaa*, *babaa*, and one other string depending on whether a context that is rewritten and then replaced by further rewriting counts as the same context (*abbaa*) or not (*abaaa*), because (with neither condition in force) $a\underline{b}\underline{b}\underline{a}a \longrightarrow a\underline{b}\underline{a}\underline{a}\underline{a} \longrightarrow ab\underline{b}\underline{b}\underline{a} \longrightarrow a\underline{b}\underline{b}\underline{a}\underline{a} \longrightarrow abaaa$. With neither condition in force, the system produces *aabaa* and *babaa* from *baaaa*.

It is interesting that Kaplan and Kay (1994), in their improved presentation of Johnson's 1970 result, present many examples where ϕ and ψ are sets of larger cardinality than 1, but not even one where they contain a string of greater length than 1. String length is essential to our understanding of the effects of Nīlakaṇṭhadīkṣitar's condition on contextualized replacement systems.

5 Replacement string length in the Aṣṭādhyāyī

The rules of the Aṣṭādhyāyī use input and output strings of length greater than 1, but it is clear that these sequences can and often do have derivational histories attached to them, i.e., not just any matching sequence will actually serve as a redex for the given rule. The English translations provided below are based upon those given in Sharma (2003).

5.1 Input

- 6.1.84: *ekaḥ pūrvaparayoḥ*, "[when saṃhitā obtains,] one comes in place of both the preceding and following." The Sanskrit *pūrvaparayoḥ* here refers to a sequence of two contiguous elements as redexes (*sthāni*) simultaneously (*yugapat*). The presence of *ekaḥ* implies that the alternative, in which there are two separate replacements of the preceding and following sounds, respectively, admits the possibility of either one being blocked independently; cf. Aṣṭādhyāyī 8.2.42 in which:

  $$t \longrightarrow n \;/\; rd \_\_,$$

  but the preceding *d* can nevertheless be replaced with *n*.

  Saṃhitā here means that the articulation of the sounds in question is closely spaced in time, defined by the traditional commentators as a pause length (*kāla*) of no more than half of a syllabic mora (*ardha-mātrā*).

- 6.1.85: "simultaneous replacement of two sounds in saṃhitā will be treated as both a final of the preceding context and an initial of the following context," e.g.:

  $$khaṭvā + indraḥ \longrightarrow [khaṭv{<}e]ndraḥ{>}$$

  This elaborates upon the how the simultaneous replacements of 6.1.84 are treated with respect to their derivational histories.

5.2 Output

Perhaps the clearest examples of these are the optional gemination rules of Aṣṭādhyāyī 8.4.46 and 47:

- 8.4.46: "A sound denoted by [the non-terminal] *yaR*, when occurring in close proximity after a vowel followed by *r* and *h*, is optionally replaced with two," e.g. $arka \longrightarrow arkka$. Perhaps this one can apply to itself (*arkkka?*): the rule states no constraints on the context that follows the duplication, but we are unable to think of an occasion when this rule would apply to a consonant that does not immediately precede a vowel.

- 8.4.47: "A sound denoted by *yaR* and occurring after [a vowel] is, optionally, replaced with two, even when [a vowel] does not follow," e.g. $dadhy{+}atra \longrightarrow daddhy{+}atra$, in which gemination of *dh* is licensed in part by the following *y*. This

one only applies to itself in the case of gemination of *y, v, r* or *l* in the so-called *paryudāsa* reading of the Sanskrit word *anacaḥ*, in which it is not translated as 'non-vowel' but rather as 'not quite a vowel.' On the other hand, Nīlakaṇṭhadīkṣitar's condition has been cited as the reason that $atra \longrightarrow attra \not\rightarrow attttra$ is blocked (Joshi and Kiparsky, 1979) under the more literal 'non-vowel' reading of this word.

- 8.1.1: "Two occur in place of one whole form . . . " This is an *adhikāra* (meta-rule) that takes scope over the next 14 rules, which license the repetition of a word or certain prefixes under specific circumstances. Sharma (2003) debates whether the repetition (*āmreḍita*) of a word that results from this rule has come about through a process of "1 $\longrightarrow$ 2" (a single instance rewrites to two instances) or a special process of "repetition of a single word." The traditional commentator, Kāśikā, says "1 $\longrightarrow$ 2," largely on the basis of how the genitive case in "of one whole form" (*sarvasya*) must be interpreted. *āmreḍita* refers here to the second of two repeated words, not consonants. It definitely cannot refer simply to the last (*param*) instance of several repeated forms because of Aṣṭādhyāyī 6.1.99, wherein we must know that the repetition was the result of an *āmreḍita* with respect to meaning (*artha*), in order to justify exempting it from Aṣṭādhyāyī 6.1.98. This is evidence that a derivational history is somehow being maintained.

Repeated application of rules (*āvṛtti*) and derivational history are perhaps the most crucial pieces of evidence that we have for understanding the restricted use of the rewriting of long sequences in the Aṣṭādhyāyī.

6 The Generative Capacity of Contextualized Replacement Systems

Let $C(j) = \{a_1^n a_2^n ... a_j^n \mid n \geq 0\}$. The set, $\{a^n b^n \mid n \geq 0\}$, presented above, is a notational variant of $C(2)$. These are the so-called count languages.

Now consider this contextualized replacement system, which generates $C(2)$:

$$
\begin{array}{l}
S \longrightarrow A \,/\, \_\_ \\
A \longrightarrow abA \,/\, \_\_ \\
A \longrightarrow ab \,/\, \_\_ \\
ba \longrightarrow X \,/\, \_\_ \\
X \longrightarrow ab \,/\, \_\_
\end{array}
$$

There are analogous systems for every other $C(j)$. They rely on cycles in derivations, but they never violate Nīlakaṇṭhadīkṣitar's condition, at least under the formal interpretation in which an empty context is in fact no context, and is therefore not counted as a previously used context. So the Pāṇinian language class includes all of the count languages.

In the years since Chomsky (1959), many language classes have been added to the Chomsky hierarchy. Some well-known ones are k-MCFL, which are generated by k CFGs in parallel, and MCTAL(k), which are generated by k parallel tree-adjoining grammars. These all lie between CFL (= 1-MCFL) and CSL, and form chains ordered by their parameter k, e.g. $1-\text{MCFL} \subset 2-\text{MCFL} \subset 3-\text{MCFL} \subset \ldots$

Each k-MCFL recognizes $C(j)$ for all $j \leq 2k$ and no more. Each MCTAG(k) recognizes $C(j)$ for all $j \leq 4k$ and no more. So the class of Pāṇinian languages is very, very strong. On the other hand, Pāṇini himself uses this power very sparingly in his grammar.

7 Concluding Remarks

The underlying formalism to Pāṇinian grammar, while our knowledge of it is incomplete, presents enough evidence to conclusively demonstrate that it is far greater in its expressive power than either RL or CFL. Pāṇini has nevertheless anticipated modern generative-syntactic practice in defining for himself a very versatile tool which he then applies very thriftily to advance his own objectives of grammatical brevity and elegance. As a result, his Aṣṭādhyāyī may even be amenable to an RL-style analysis, as Hyman (2007) has claimed. But in light of this investigation, the result of this analysis certainly could not be a grammar in Pāṇini's own style, but rather Pāṇini's grammar recast into someone else's style.

We have not even touched upon perhaps the greatest difference between Pāṇini's own formalism and the standard string-rewriting systems concomitant with Chomsky's hierarchy, which is its built-in capacity for disambiguation. Pāṇini's grammar, through its use of rule precedence and other meta-conventions, generates a single derivation for every grammatical sentence of Sanskrit.

Not even a single one of the standard Chomskyan systems possesses this property, and it is this lack of theirs, rather than some inherent quality of the syntax of human languages that is responsible for the now-widespread use of numerical reasoning and statistical pattern recognition methods in natural language processing. These are required in order to curb the natural propensity of these algebras to overgenerate. Through the lens of contemporary NLP, the most amazing fact about the Aṣṭādhyāyī is not that it produces so many correct derivations, after all, but that it simultaneously avoids so many incorrect ones.

References

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2:137–167.

Culy, C. (1985). The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, 8:345–351.

Huybregts, M. A. C. (1984). The weak adequacy of context-free phrase structure grammar. In de Haan, G. J., Trommelen, M., and Zonneveld, W., editors, *Van periferie naar kern*, pages 81–99. Foris.

Hyman, M. D. (2007). From Pāṇinian sandhi to finite state calculus. In Huet, G. and Kulkarni, A., editors, *Proceedings of the First International Symposium on Sanskrit Computational Linguistics*, pages 13–21.

Ingerman, P. Z. (1967). "Pāṇini-Backus Form" suggested. *Communications of the ACM*, 10(3):137. Letter to the Editor.

Johnson, C. D. (1970). *Formal Aspects of Phonological Description*. PhD thesis, University of California, Berkeley.

Joshi, S. D. and Kiparsky, P. (1979). Siddha and asiddha in Pāṇinian phonology. In Dinnsen, D., ed., *Current Approaches to Phonological Theory*, pages 223–250. Indiana University Press.

Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Sharma, R. N. (1987–2003). *The Astadhyayi of Panini*, volume I–VI. Munshiram Manoharlal Publishers Pvt. Ltd.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–344.

Staal, F. (1965). Context-sensitive rules in Pāṇini. *Foundations of Language*, 1:63–72.