John J. Lowe

Revisiting Pāṇini's generative power

Abstract: In this paper I revisit recent claims that Pāṇini's generative grammar has fully context-sensitive power. I show that in fact Pāṇini assumed acyclicity as a constraint on rule application, limiting the power of his superficially context-sensitive formalism in a way entirely parallel to much work in modern phonology.

Keywords: Generative power, Panini, acyclicity, constraint rules

1 Introduction

Pāṇini's *Aṣṭādhyāyī* has been hailed as the world's first generative grammar, composed more than 2,000 years before the advent of modern generative linguistics.[1] From the very beginnings of modern generative grammar, the questions of a grammar's generative power, and of the generative power required to model human language, have been central in the development of grammatical formalisms and in the evolution of the field.[2] A number of authors have analyzed Pāṇini's *Aṣṭādhyāyī* in these terms, including Staal (1965, 1966), Hyman (2007) and Penn and Kiparsky (2012). Pāṇini's *Aṣṭādhyāyī* is superficially a context-sensitive grammar, though this observation does not in itself answer the question of its generative power.[3] While Hyman (2007) claims that Pāṇini's system has only the power of a finite state transducer, Penn and Kiparsky (2012) seek to prove that

1 Chomsky (1965: v): "it seems that even Pāṇini's grammar can be interpreted as a fragment of such a "generative grammar," in essentially the contemporary sense of this term".

2 The term "modern generative grammar" encompasses a number of theories of grammar, from that of Chomsky's *Syntactic Structures* to the relative plethora of more modern theories such as Government & Binding Theory, the Minimalist Program, Lexical-Functional Grammar, Head-driven Phrase Structure Grammar, Sign-Based Construction Grammar etc.

3 The focus here is on weak generative capacity. The driving principle behind Pāṇini's grammar is not the search for a theory of linguistic competence but the desire for the most concise and efficient description capable of generating all and only the grammatical utterances of Sanskrit; questions of strong generative capacity are therefore anachronistic to an analysis of the *Aṣṭādhyāyī*.

John Lowe, University of Oxford, The Oriental Institute, Pusey Lane, OX1 2LE, Oxford, UK. Email: john.lowe@orinst.ox.ac.uk

https://doi.org/10.1515/9783110753066-014

Pāṇini's formalism has the full power of a context-sensitive grammar (i.e. the power to generate all the context-sensitive languages). This paper revisits the claims of Penn and Kiparsky (2012), and reveals flaws in their argument. Pāṇini in fact assumed acyclicity as a constraint on rule application, significantly restricting the generative power of his formalism.

In section 1, I provide an overview of the issue of generative power of grammatical systems. In section 2, I discuss previous accounts of Pāṇini's generative power. In sections 3–4, I revisit the arguments of Penn and Kiparsky (2012) and show that Pāṇini assumed acyclicity.

2 Formal language theory and generative power

Chomsky (1957: 13) defined a *language* as: "a set (finite or infinite) of sentences, each finite in length and constructed out of a finite set of elements". Given this definition, the goal of linguistic analysis of a given language L is "to separate the *grammatical* sequences which are sentences of L from the *ungrammatical* sequences which are not sentences of L and to study the structure of the grammatical sequences. The grammar of L will thus be a device that generates all of the grammatical sequences of L and none of the ungrammatical ones" (Chomsky 1957: 13). Pāṇini's *Aṣṭādhyāyī* generates, or licenses the derivation of, all and only the possible grammatical sentences of Sanskrit, and in this crucial sense it can be understood as a generative grammar of the language.

Languages may have different levels of complexity, and may correspondingly require more or less complex grammars to generate them. In relation to the linguistic study of human languages, an important theoretical question is how complex human languages are, or can be. Related to this is a separate question: how complex must or should a grammar (or set of grammars licensed by a particular grammatical theory) be in order to both adequately and insightfully model human language?

The so-called "Chomsky Hierarchy" (Table 1) distinguishes four types of languages according to their computational complexity (based on Chomsky 1959): Type 0, the recursively enumerable or unrestricted languages; Type 1, the context-sensitive languages; Type 2, the context-free languages; Type 3, the regular languages. The type of grammars required to generate these languages differ in terms of the complexity of rules permitted in the grammars (or to look at it

another way, in terms of the constraints placed on rule formulation in the grammars).[4]

Tab. 1: The Chomsky Hierarchy

Language class: grammar	Rule format	Conditions on rules
Type 0: Unrestricted grammars	A→B	A, B ∈ $[V_T \cup V_N]^*$
Type 1: context-sensitive grammars	A→B / C __ D (or CAB→ CBD)	A ∈ V_N B ∈ $[V_T \cup V_N]^+$ C,D ∈ $[V_T \cup V_N]^*$
Type 2: Context-free grammars	A→B	A ∈ V_N B ∈ $[V_T \cup V_N]^+$
Type 3: Regular grammars	A→a B	A ∈ V_N B ∈ $[V_N \cup \{\}]$ a ∈ V_T

Each class of languages includes those classes below it on the hierarchy, such that Type 0 ⊃ Type 1 ⊃ Type 2 ⊃ Type 3. This means that regular languages also fall in the class of context-free languages, as well as in the classes of context-sensitive and unrestricted languages; context-free languages also fall in the class of context-sensitive and unrestricted languages, etc. But there are context-free languages which are not regular languages, context-sensitive languages which are not context-free languages, and context-sensitive languages which are not unrestricted languages. Likewise, a context-free, context-sensitive or unrestricted grammar could be written to generate any regular language, but a regular grammar can only generate regular languages. The grammars in the Chomsky Hierarchy correspond to machines of differing complexity: regular languages can be computed by finite state machines; context-free languages can be computed using push-down store automata; context-sensitive languages can be computed using linear bounded automata; and unrestricted languages can be computed using Turing machines.

Formal language theory investigates the properties of grammars by abstracting away from the messy reality of human languages and focusing on strings of abstract symbols which have a precise mathematical complexity. Consider a language to be a set of sentences, and consider a sentence to be a string of symbols (words, terminals). Then we can take sets of strings of abstract symbols, analyze

4 In (1), V_T refers to the set of terminal elements in a grammar, and V_N to the set of non-terminals.

their mathematical complexity, and analyze the complexity required of grammars to model them. For example, the following regular (right linear) grammar models the language $\{a^n b^m \mid n, m \geq 1\}$:

(1) — Start symbol: A
— Non-terminals: {A, B}
— Terminals: {a, b}
— Rules:
— A→ a A
— A→ a B
— B→ b B
— B→ b

The language $\{a^n b^m \mid n, m \geq 1\}$ consists of all the strings which contain one or more *a*s followed by one or more *b*s, so: *ab*, *aab*, *aaab*, *aaaab*... *abb*, *aabb*, *aaabb*..., *abbb*...etc. The number of grammatical strings in this language is infinite, and the regular grammar in (1) generates all and only these strings.

The language $\{a^n b^n \mid n \geq 1\}$ is too complex for a regular grammar. This language consists of all the strings which contain one or more *a*s followed by the same number of *b*s. This language also contains an infinite number of grammatical strings, though a subset of the strings in the language $\{a^n b^m \mid n, m \geq 1\}$. The additional complexity introduced by requiring that strings have the same number of *a*s and *b*s necessitates a context-free grammar:

(2) — Start symbol: A
— Non-terminals: {A}
— Terminals: {a, b}
— Rules:
— A→ a A b
— A→ a b

To model the language $\{a^n b^n c^n \mid n \geq 1\}$, we require a context-sensitive grammar:

(3) — Start symbol: A
— Non-terminals: {A,B}
— Terminals: {a, b, c}
— Rules:
— A→ abc / __
— A→ aAB / __

— cB→ Bc / __
— B → bc / b__c

The Chomsky Hierarchy and the question of what kind of grammar is required, or desired, to model human language, has had a significant impact on the development of modern grammatical theory. Following Chomsky (1957), it was long taken as evident that human languages are too complex to be modelled using a context-free grammar. Chomsky's initial attempt to formulate a richer model resulted in "transformational" grammar, with equivalent power to an unrestricted grammar. Pullum and Gazdar (1982) showed that most of the phenomena previously taken as evidence that human language is more than context-free could, in fact, be given a context-free treatment. This led to the development of Generalized Phrase Structure Grammar (Gazdar et al. 1985), a formal model of language explicitly restricted to context-free power. Subsequently, Huybregts (1984) and Shieber (1985) proved that Swiss German is not syntactically context-free but requires some context-sensitive power to account for cross-serial dependencies, and Culy (1985) showed that Bambara is morphologically more than context-free.

It appears, then, that human language is *mildly context-sensitive*: some power beyond context-free is required to model human language in general, but only a little, and not necessarily for all languages. Based on this observation, some authors sought to develop formalisms that are mildly context-sensitive, i.e. that can generate some but not all context-sensitive languages, and that are just a little more complex than context-free grammars, but not so complex as to be computationally intractable. Such formalisms include multiple component tree-adjoining grammars (Joshi et al. 1975) and multiple context-free grammars (Seki et al. 1991, Clark 2015).[5]

While one line of development therefore sought to restrict the power of grammatical formalism to the minimum required for modelling human language, the other approach taken was to use a more powerful formalism, but to state constraints within the formalism which limit the power of any individual grammar to a tractable level. This is the approach taken in frameworks such as Minimalism, Lexical-Functional Grammar and Head-driven Phrase Structure Grammar; Pollard (1996) points out that this is the standard approach in other sciences.[6]

5 These are relevant in Penn and Kiparsky's (2012) account of Pāṇini's generative power, discussed below.

6 See also Müller (2016: 529–533).

Generative power is therefore a considerably more complicated issue than the Chomsky Hierarchy itself suggests.[7] For example, if a context-sensitive grammar is permitted to include deletion rules (or if {} is included in the set of terminal symbols), then that grammar has the power of an unrestricted grammar (Coleman 1991: 128–130). From the other side, if deletion rules are removed from an unrestricted grammar, that grammar has at most the power of a context-sensitive grammar.

Constraints on rule application can also restrict the power of a grammar. As shown by Johnson (1972), if the rules of a context-sensitive grammar are prevented from applying cyclically to their own output ('acyclicity'), the grammar has only finite-state power, i.e. it can generate only the regular languages.[8] Notice that the grammar in (3) can only generate $\{a^n b^n c^n\}$ by allowing the rule A → aAB to apply cyclically to its own output; if we enforce cyclicity, the grammar can generate only the two strings *abc* and *aabbcc*. This will become important in what follows.

3 The *Aṣṭādhyāyī* as a formal system

3.1 The context-sensitive surface

Presentationally, at least, Pāṇini's grammar is context-sensitive: rules can be formulated with reference to a preceding and/or following context. Within the grammar, the Sanskrit case system is appropriated for technical purposes: cases are invested with technical functions (in addition to their ordinary uses, where required):

(4) — Nominative: an element added, or which substitutes for another element.
— Genitive: an element which is replaced by the element given in the nominative.
— Ablative: the preceding context for the operation.
— Locative: the following context for the operation.

7 Coleman (1995: 333–340) has a very clear and useful summary of generative power; see also Kaplan and Kay (1994).

8 See also Kaplan and Kay (1994: 346), who state the restriction thus: "the part of the string that is actually rewritten by a rule is excluded from further rewriting by that same rule".

In schematic terms, we can reformulate Pāṇini's system into the familiar modern context-sensitive representation in the following way:[9]

(5) $A_{gen} \rightarrow B_{nom} / C_{abl} \_\_ D_{loc}$

By way of illustration, consider the following sandhi rule:

(6) *ik-o* *yaṇ* *ac-i* (72: *saṃhitāyām*)
iK-GEN yaṆ.NOM aC-LOC connection.LOC
'In connected speech a sound of the group denoted by iK (i.e. *i*, *u*, *ṛ*, *ḷ*) is replaced by the corresponding sound of the group denoted by yaṆ (i.e. *y*, *v*, *r*, *l*) when one of the sounds denoted by aC (i.e. a vowel) follows.' (Aṣṭ. 6.1.77)

This is the rule that converts final vowels *i*, *u*, *ṛ*, *ḷ*[10] into corresponding semi-vowels in context before a vowel; e.g. *dadhi+atra* '(there is) yoghurt here' surfaces as *dadhy atra*. The following context is clearly indicated by the locative case marking on the pratyāhāra *a*C.[11]

But as we have seen in the preceding section, knowing that Pāṇini's *Aṣṭādhyāyī* is a context-sensitive grammar is insufficient, in itself, for determining its generative power. The *Aṣṭādhyāyī* does permit deletion, so it could, without further constraint, have unrestricted power. But Hyman (2007), focusing on the sandhi rules of the *Aṣṭādhyāyī*, claims that rules do not rewrite their own output, and thus the sandhi portion of the *Aṣṭādhyāyī*, at least, can be reduced to a regular grammar.[12]

9 For formal discussion of translating Pāṇinian replacement rules into standard CS representations as we are familiar with, see Staal (1965) and Cardona (1965).

10 *ḷ* vacuously, since this never occurs word finally.

11 Pratyāhāras are sequences of a grammatical element followed by a code letter (anubandha or 'it' – rendered in small caps), which serve to refer to sets of grammatical elements in the grammar, as iK in (6) refers to the segments *i*, *u*, *ṛ* and *ḷ*.

12 A slightly different association is sometimes made, between Pāṇini's formalism and context-free systems, in particular the 'Backus-Naur Form' which was centrally significant in the development of the historically important computer languages ALGOL58 (Backus 1959) and ALGOL60 (Backus et al. 1963). The Backus-Naur Form is a notation for context-free grammars, and the principles behind it were associated with the principles of Pāṇini's formalism by Ingerman (1967). However, the similarities relate specifically to the use of pratyāhāras to denote groups of symbols, and do not extend to the generative workings of the *Aṣṭādhyāyī* itself.

Joshi and Kiparsky (1979: 244–245) show that rules are permitted to apply cyclically in the sense that rules may apply multiple times in a given derivation. However, there are constraints on the reapplication of rules. Joshi and Kiparsky (1979: 244) refer to a paribhāṣā, an interpretative rule, attributed to Nīlakaṇṭhadīkṣita, which they understand to mean that "a rule cannot be conditioned twice in a derivation by the same context". Penn and Kiparsky (2012) refer to this as "Nīlakaṇṭhadīkṣitar's [sic] condition", and seek to show that it is a considerably weaker constraint than acyclicity, and does not itself reduce the power of the *Aṣṭādhyāyī* below full context-sensitivity.

3.2 Penn & Kiparsky's argument

As discussed in more detail below, Joshi and Kiparsky (1979) argue that Pāṇini presupposed a constraint on rule application to the effect that a rule may not reapply given the same (partial or complete) context as a preceding application, but that Pāṇini's system does not presuppose acyclicity, since Pāṇini states ad hoc constraints to ensure this when the constraint on reuse of contexts is not enough. Penn and Kiparsky (2012) accept this interpretation of Pāṇini's rules; below, I show that this interpretation is in fact problematic. For the time being, however, let us also adopt this interpretation, for the sake of argument, and investigate Penn and Kiparsky's claims.

Penn and Kiparsky (2012) begin by demonstrating that the constraint on reuse of contexts (their "Nīlakaṇṭhadīkṣitar's condition") is weaker than a full constraint on cyclicity. I here provide a slightly simpler demonstration than that provided by Penn and Kiparsky, using the following grammar:[13]

(7) 1. *aa* → *bb* / b__a
2. *b* → *a* / b__a

Given the input *baaaa*, we have the following derivations. Rule 1 must apply first, giving *bbbaa*: the context for application of this rule in this case is the first and the fourth symbols, and the elements rewritten are the second and third symbols. Rule 2 must then apply, with the context being the second and fourth symbols, and the element rewritten being the third. This gives *bbaaa*. If we assume acyclicity, i.e. that a rule may not rewrite (all or part of) its own output, Rule 1 cannot

13 This grammar is a reduced form of that provided by Penn and Kiparsky (2012), including only the first two rules.

be applied to the string *bbaaa*, since it would rewrite the third symbol, which was already rewritten by Rule 1 in its first application. Rule 2 can reapply, however, giving *baaaa*, and no further rule applications are possible, meaning that this grammar, assuming acyclicity, takes *baaaa* and returns only *baaaa*.

However, if we assume merely a constraint on reuse of contexts, there is another possibility at the stage *bbaaa*. Rule 1 may be reapplied instead of Rule 2, because the context is different from its first application: now the context of application of Rule 1 is the second and the fifth symbols. This gives *bbbba*. Rule 2 will then apply, with context of third and fifth symbols, to give *bbbaa*. This is the end of the derivation, because *bbbaa* does not provide a valid context for Rule 1, and Rule 2 has already applied to the string *bbbaa*, meaning that it cannot reapply without reusing the context used previously.

The constraint on reuse of contexts therefore outputs two strings given the input *baaaa*, and the grammar is therefore less constrained than if subject to acyclicity, which returns only one string. If neither constraint is applied, the grammar licenses infinite derivation chains.

Penn and Kiparsky (2012) try to demonstrate that Pāṇini's system, presupposing the constraint on reuse of contexts, has fully context-sensitive power. They provide a context-sensitive grammar which they say obeys the principles of Pāṇini's system, and which can model the language $\{a^n b^n\}$; given the formulation of this grammar, it can unproblematically be augmented to model any language defined by $C(j) = \{a_1{}^n a_2{}^n \ldots a_j{}^n | n \geq 0\}$, i.e. also $\{a^n b^n c^n\}$, $\{a^n b^n c^n d^n\}$, etc. The grammar they use is:

(8)
- S → A / __
- A → abA / __
- A → ab / __
- ba → X / __
- X → ab / __

Essentially this is a way of generating $\{a^n b^n\}$ by producing arbitrarily long strings of the form *ab*, *abab*, *ababab*, *abababab* etc., and then flipping sequences of *ba* to give *ab*, until all *a*s precede all *b*s. The same thing can easily be done with the same basic structure for $\{a^n b^n c^n\}$, $\{a^n b^n c^n d^n\}$, etc.

Penn and Kiparsky (2012) relate this generative potential to the multiple context-free languages and the multicomponent tree adjoining grammars. These lie in generative power between context-free power and full context-sensitive power. For example, a multiple context-free grammar *k*-MCFG is generated by *k* context-free grammars in parallel, and can generate $C(j)$ for all $j \leq 2k$, such that 1-MCFG is

equivalent to a context-free grammar, and can therefore generate $\{a^n b^n\}$, while 2-MCFG is a mildly context-sensitive grammar and can generate $\{a^n b^n\}$, $\{a^n b^n c^n\}$, $\{a^n b^n c^n d^n\}$. Since the grammar given in (8) can generate *C(j)* for any *j*, it is as powerful as the most powerful MCFG, i.e. it has fully context-sensitive power. Penn and Kiparsky (2012) therefore claim that Pāṇini's system has very strong generative power, although the *Aṣṭādhyāyī* itself uses only a fraction of this power.

There are a number of problems with these claims of Penn and Kiparsky (2012). I will investigate these in the next section.

4 Revisiting Pāṇini's power

The first problematic aspect of Penn and Kiparsky's argument is as follows. They argue that the grammar in (8) does not violate the condition against reusing contexts, on the assumption that the null context is no context and therefore does not count. But the rule:

(9) $A \rightarrow abA \; / \; \_$

is not really a null context rule, at least if we are taking the constraint on reuse of contexts seriously. It could be rewritten as:

(10) $\epsilon \rightarrow ab \; / \; \_A$

and then it could only be reapplied by taking the same context repeatedly. To do it otherwise undermines the whole point of the constraint in question: as phrased in (9), Penn and Kiparsky (2012) nullify any restrictive power that the constraint on reuse of contexts might have, by putting the context into the input/output and assuming that a null context is no context. But if we phrase rules in such a way that a constraint on reuse of contexts can actually have an effect, then we will have to phrase as in (10), and such rules will not be able to apply multiple times in the same context.

Almost all the rules in (8) also depend on another assumption crucial to Penn and Kiparsky's argument: that Pāṇini's system permits input and output strings of length greater than 1. Three of the rules in (8) have output length greater than 1 (once 3, twice 2), and the fourth rule has input length of 2. To generate more complex languages would require more: for example, for $\{a^n b^n c^n d^n\}$ we would

require a rule with output of length 5, in Penn and Kiparsky's formulation.[14] It is true that Pāṇini does technically allow input and output greater than length 1, but in practice this is highly restricted, and this practical restriction could reflect some restrictions in principle. The majority of cases of output greater than length 1 are cases of doubling: where one element in the input produces two instances of itself in the output. Other cases are much rarer, though they do occur: for example, affix insertion or replacement may be specified with concomitant insertion of a second element (an *āgama* 'augment'), as in the derivation of names like *indrāṇī* (Aṣṭ. 4.1.49). Going beyond two elements in the output is highly unusual and arguable; one such example, discussed in the ancient tradition for its complexity, is rule 6.4.47, which deletes the *r* and *s* segments from the root *bhrasj* and inserts *ra*M at the same time, i.e. three operations.[15] Now these are limitations on the actual rules formulated by Pāṇini, rather than necessary constraints on the kinds of rules that he could have formulated, but at least to some extent we can only determine the restrictions on rule formulations that Pāṇini assumed on the basis of the actual rules he formulated. Pāṇini may in fact have assumed an absolute constraint on output length of 3, for example.

Given these facts, it is clear that the grammar given in (8) is less constrained than Pāṇini's system, and its generative power will be significantly restricted if we take seriously the constraint on reuse of contexts and interpret the rule A → abA / _ as meaning $\epsilon \rightarrow$ ab / _A. If we admit output of length 3, then it would be possible to generate the language $\{a^n b^n\}$ in Pāṇini's system, using something like the context-free grammar in (2) above. But note that this is only possible if we accept "Nīlakaṇṭhadīkṣitar's condition" as formulated by Penn and Kiparsky (2012). If it were the case that Pāṇini instead assumed acyclicity, even $\{a^n b^n\}$ would be ruled out. Penn and Kiparsky's understanding of Nīlakaṇṭha's constraint on rule application is based on Joshi and Kiparsky (1979: 244). In the next section, I revisit their claims regarding the inferences possible from Pāṇini's rules, as well as revisiting the formulation of the constraint itself.

14 I.e. A→ abcdA / _.

15 This is if we assume the input is the sequence of five segments bh-r-a-s-j. Alternatively, we might consider this a morphological rule affecting the single morphological element *bhrasj*, effectively replacing *bhrasj* with *bhaj* and inserting *ra*M, so only two operations.

5 Revisiting the constraint

As discussed, Joshi and Kiparsky (1979: 244) argue that Pāṇini assumed a constraint on rule application to the effect that "a rule cannot be conditioned twice in a derivation by the same context". They seek to demonstrate that this constraint was assumed by Pāṇini with two complementary arguments: in some cases the constraint must be assumed in order to get the right output, and in other cases Pāṇini states rules to explicitly prevent rules from re-writing their own output (i.e. to prevent cyclicity), when the assumed constraint on reuse of contexts is insufficient to guarantee this. In other words, Pāṇini uses soft constraints (rules within the grammar) to ensure acyclicity where necessary, but the presupposed constraint on rule reapplication (that "a rule cannot be conditioned twice in a derivation by the same context") is weaker.

To first illustrate multiple rule application, Joshi and Kiparsky (1979: 244) cite the form *bhūyāt*, 3sg. precative, which derives from *bhū+yās+st*, to which a rule applies which deletes *-s-* at the start of word final clusters or clusters before obstruents (8.2.29 *skoḥ saṃ yogādyor ante ca*). This rule, which has to apply twice, states that 'there is deletion of *s* or *k* which begin a cluster before a stop or fricative sound or at the end of a word'. So the first deletion affects the first *s*, because it is at the start of a cluster (*ss*) which precedes another obstruent. Then the second *s* is deleted because it is at the start of a cluster (*st*) which ends a word. So there are two applications of the same rule, but they do not have the same context.[16]

To illustrate the assumed constraint on reuse of contexts, Joshi and Kiparsky (1979: 245) cite rule 8.4.47, which I give along with the preceding rule, since this is also relevant:

(11) 8.4.46: *acaḥ rahābhyāṃ dve* 'All consonants except /h/ following an /ɾ/ or /h/ following a vowel, are optionally (8.4.45) doubled.'
(12) 8.4.47: *anaci ca* 'All consonants except /h/ after a vowel and before a non-vowel are optionally doubled.'

In reference to 8.4.47, Joshi and Kiparsky (1979: 245) cite the word *atra* 'here', to which this rule may apply to give *attra*. They state that since the rule must not be allowed to reapply, to give **atttra*, **attttra* etc., this means that the left-hand context "after a vowel" cannot be reused. Note that this is only a partial context, since

16 I show below that these two applications still respect cyclicity, but for now I present Joshi and Kiparsky's argument.

if the rule were to reapply, the right-hand context (the 'non-vowel') would be different in each iteration.

In fact this is not an optimal example, because there is another rule which can delete a stop or sibilant following a consonant and before a homophonous sound:

(13) 8.4.65: *jhaRaḥ jhaRi savarṇe* 'There is optional deletion of a stop or sibilant before a homophonous stop or sibilant and after a consonant (*halaḥ* inferred from 64).'

If 8.4.47 could apply multiple times in the same partial context, then 8.4.65 could also apply multiple times in the same partial context to effectively nullify all but one of the applications of 8.4.47.[17] Nevertheless, we still require the constraint on reuse of contexts, because 8.4.46–7 includes more consonants in its scope than 8.4.65. Semi-vowels and nasals do not fall under the scope of 8.4.65, but can be doubled by 8.4.46–7. So, the word *avyaya* 'unchanging, indeclinable' can be doubled by 8.4.47 to *avvyaya*, but the potentially infinite recursive reapplication of the same rule must be prevented. Similarly, 8.4.46 permits a word like *brahman* 'Veda, sacred text, knowledge' to be pronounced *brahmman*, but reapplication of the rule to produce **brahmmman*, **brahmmmman*, etc. is undesirable. Examples such as these therefore make a constraint on reuse of contexts both viable and necessary.[18] Notice that they would also, however, support a stronger constraint, i.e. acyclicity, which would have the same effect.

The second example that Joshi and Kiparsky (1979: 245) give is slightly different, and more problematic. The rule 7.4.59 *hrasvaḥ* specifies that a short vowel replaces the vowel of a reduplication syllable. The effect of this rule is therefore to shorten any long vowels found in reduplication syllables. In the formation of the perfect of *āp*, we have the following derivation sequence:

(14a) *āp+āp+a*
(14b) *ap+āp+a* (7.4.59)

17 Assuming an intelligent user, this would license the correct output, but since 8.4.65 is optional (as are 8.4.46–7), from a purely mechanical perspective it would be possible to derive outputs such as **atttra*, **attttra* etc., or even infinitely long derivation chains. This possibility does therefore need to be ruled out.

18 The need to prevent multiple applications of 8.4.46–7 was recognized and discussed by the later grammatical tradition, in the context of the *paribhāṣā* discussed in this section. See also fn. 23 below.

(14c) *a+āp+a* (7.4.60)
(14d) *āp+a* (6.1.101)

The first step shows the duplicated root followed by the 3sg. perfect ending *-a*. The first occurrence of *āp* here is the reduplication syllable, and the second is the original root syllable. In (15b) the shortening rule 7.4.59 applies to the reduplication syllable to give *ap-āp-a*. 7.4.60 then applies to delete the final consonant of the reduplication syllable (15c), and vowel coalescence then gives *āp-a* (15d). The resulting element *āp* technically counts as both the reduplication syllable and the stem (by 6.1.85). Crucially, 7.4.59 does not then reapply to the reduplication syllable to give **apa*.

Joshi and Kiparsky (1979: 245) take this as another example of the constraint on reuse of contexts, but it is not clear what the context is here. Joshi and Kiparsky seem to assume that the context is 'in case of a reduplication syllable', but in fact this is not the context but the input to the rule. It is more appropriate to understand the non-reapplication of 7.4.59 as due to a constraint preventing cyclicity: 7.4.59 takes a reduplication syllable as input, and outputs the same syllable but with a short vowel. The same reduplication syllable cannot then be input to this rule again.

The other crucial part of Joshi and Kiparsky's argument is that Pāṇini explicitly prevents cyclicity where the constraint on reuse of contexts is insufficient. This provides crucial evidence, according to their argument, that Pāṇini must have presupposed a constraint on reuse of contexts but not a constraint on cyclicity. To argue this, Joshi and Kiparsky (1979: 245) discuss the derivation of another perfect form, *vivyādha*, perfect of *vyadh* 'pierce':

(15a) *vyadh+vyādh+a*
(15b) *viadh+vyādh+a* (6.1.17)
(15c) *vidh+vyādh+a* (6.1.108)
(15d) *vi+vyādh+a* (7.4.60)

This is partly parallel to the previous example, but here the form of the reduplication syllable is initially different, and has to undergo different alterations. Rule 6.1.17 causes vocalization of the semivowel in the reduplication syllable, giving *viadh-* (16b), which reduces to *vidh-* by 6.1.108 (16c), and to *vi-* by 7.4.60 (16d). Now we do not want the semivowel vocalization rule to reapply to *vi-*, to give something like **uivyādha*. In this case, Pāṇini states a rule (6.1.37 *na saṃprasāraṇe saṃprasāraṇam*) which prevents vocalization applying to a semivowel which precedes a vocalized semivowel. Joshi and Kiparsky (1979: 245)

claim that this provides evidence that Pāṇini did not assume acyclicity, but has to enforce it here because the constraint on reuse of contexts would not by itself prevent reapplication of 6.1.17.

We have seen already, however, that the other examples provided by Joshi and Kiparsky (1979: 244–245) in support of "Nīlakaṇṭhadīkṣitar's constraint" can also be understood if we take the constraint as a full constraint on cyclicity. In this case, the question is the relevance or redundancy of 6.1.37. If Pāṇini assumed acyclicity, then the need for 6.1.37 is not immediately obvious, because acyclicity is sufficient to prevent reapplication of 6.1.17 to the same reduplication syllable. Must we therefore assume that Pāṇini adopted a weaker constraint, such as a constraint on reuse of contexts? Not necessarily. The necessity of 6.1.37 had already been considered by the ancient grammatical tradition; the details of the issues involved go beyond the scope of this paper, but 6.1.37 is justified on the assumption that 6.1.17 would otherwise apply to all relevant semivowels in the input.[19] That is, 6.1.37 does not enforce acyclicity, but simply restricts the single application of 6.1.17 such that it affects only the second of two contiguous semivowels which come into its scope.

To return to the very first example given in this section, *bhūyāt*: Joshi and Kiparsky (1979: 244) gave this as an example of multiple application of a rule to the same form, but in fact the inputs to the two applications of the rule are the two separate *s* segments, so the two applications do not, in fact, violate acyclicity, as Joshi and Kiparsky imply. All subsequent examples discussed by them build on the assumption that cyclic rule application was possible, but in fact everything we have seen can be better interpreted if we assume that Pāṇini presupposed not a weak constraint on reuse of contexts, but a stronger constraint on the reapplication of a rule to its own output, i.e. acyclicity.

Joshi and Kiparsky (1979) interpret the supposed constraint on reuse of contexts from Pāṇini's rules, as we have discussed, but they and Penn and Kiparsky (2012) also understand this constraint to be directly stated in a paribhāṣā, an interpretative rule, which they attribute to the 18th century grammarian Nīlakaṇṭhadīkṣita.[20] If the constraint is in fact to be interpreted in the way Joshi and Kiparsky argue it should, this would support their claims regarding the above derivations. But in fact, it is not.

The paribhāṣā in question is in fact earlier than Nīlakaṇṭha, being first attested in Puruṣottamadeva's *Paribhāṣāpāṭha* (c. 12th century); Nīlakaṇṭha provides the first attested commentary on the rule, in his *Paribhāṣāvṛtti*. For both,

19 See Sharma (2001: 45–47) for a summary of the understanding of this rule.

20 This grammarian is discussed briefly by Coward and Kunjunni Raja (1990: 373).

see Abhyankar (1967: 160b, 313).[21] In Abhyankar (1967: 313) the text is given as follows, together with Nīlakaṇṭha's three lines of commentary. I also give the text and commentary of the preceding paribhāṣā, which is directly relevant:[22]

(16) 111: *parjanyaval lakṣaṇapravṛttiḥ*.
'kṛtakāri khalv idaṃ śāstraṃ meghavat' iti iko jhal (1.2.9) iti sūtrasthabhāṣyeṇa nyāyasiddheyam. tena ūkhatur ityādau hrasvasyāpy abhyāsahrasvatve tato dīrghe punar hrasvo na bhavati. lakṣye lakṣaṇasya sakṛd eva pravṛttir iti nyāyāt.
112: *lakṣye lakṣaṇasya sakṛd eva pravṛttiḥ*.
'yathoktaviṣayeṣu hrasvāpravṛttiḥ. 'irayo re' (6.4.76) iti dvivacananirdeśo 'syā jñāpaka iti spaṣṭam 'ekaḥ pūrvaparayoḥ' (6.1.84) iti sūtre bhāṣye.'

Translation:

(17) 111: The application of a rule is like the rain.
'Rules are like the rain in effecting something that is already done.' This paribhāṣā is established by principle in the *Mahābhāṣya* on the sutra *iko jhal* (Aṣṭ. 1.2.9). By this, in the shortening of the reduplication syllable even of one that is already short, (e.g.) at the start of the word *ūkhatuḥ*, the long vowel does not again become short. This is on the basis of the principle, the application of a rule to its target happens once.'
112: The application of a rule to its target happens once.
'As in the examples stated there is no occurrence of the short vowel. The indicator of this (paribhāṣā) is the specification of dual number in *irayo re* (Aṣṭ. 6.4.76). This is clear in the *Mahābhāṣya* (commenting) on *ekaḥ pūrvaparayoḥ* (Aṣṭ. 6.1.84).'

The first paribhāṣā given here is not directly relevant to the present discussion, but introduces the second paribhāṣā with reference to yet another example involving perfect tense reduplication. The form *ūkhatuḥ* is a 3rd person dual perfect

21 The paribhāṣā in question does not however appear in Nāgeśa's *Paribhāṣenduśekhara*, although it is discussed under the *parjanyavat* paribhāṣā (see 16).

22 The text of the crucial paribhāṣā is (trivially) slightly different as given by Joshi and Kiparsky (1979: 250) and Penn and Kiparsky (2012), who adopt the phrasing of Puruṣottamadeva's *Paribhāṣāpāṭha*: *lakṣye lakṣaṇaṃ sakṛd eva pravartate*.

form, which would correspond to a 3SG. *ūkha*. Crucial here is the first syllable, which in some respects is parallel to that of *āpa*, discussed above. The difference is that the root *ukh* has a short root vowel to start with. We therefore have the following derivation of the 3SG. *ūkha*:

(18a) *ukh+ukh+a*
(18b) *ukh+ukh+a* (7.4.59)
(18c) *u+ukh+a* (7.4.60)
(18d) *ūkh+a* (6.1.101)

The derivation chain is parallel to that of *āpa* above, except that the application of 7.4.59 in (19b) is vacuous, because the vowel of the reduplication syllable is already short. This is the point of paribhāṣā 111: just as the rain falls indiscriminately on land and on water (which is already wet), so rules apply even where their effects are vacuous. The fact that 7.4.59 has already applied, albeit vacuously, at stage (19b) is what prevents it from reapplying at stage (19d). The reason given for this is the second paribhāṣā, which is the one in question here: 'the application of a rule to its target happens once'.

In terms of the phrasing of the paribhāṣā, the crucial word is *lakṣya*. The application of a rule, *lakṣaṇa*, occurs (only) once *lakṣye* 'in *lakṣya*'. Joshi and Kiparsky (1979) and Penn and Kiparsky (2012) apparently take this to mean 'in a given context', but the primary meaning of *lakṣya* in grammatical literature is 'target of a rule', i.e. the element to which a rule applies, the element which is input to a rule.[23] Thus the most natural reading of the paribhāṣā is as a statement of acyclicity, not as a constraint on reuse of contexts. The translation given in (17) reflects this.

This reading is supported by the examples given. The case of *ūkha* or *ūkhatuḥ* is exactly parallel to *āpa* above: the non-reapplication of 7.4.59 is not to do with a reuse of context, because there is no context: the rule takes as input reduplication syllables, and it is only a constraint which prevents cyclic application of a rule to the same input which will give the desired outcome here.

The second example given in Nīlakaṇṭha's commentary on paribhāṣā 112 provides the exception that proves the rule: it shows a case where Pāṇini exceptionally permits cyclic application of a rule to its own output, something that would

23 This is not the only possible sense of *lakṣya*, but the range of this word generally encompasses the element, form or word to which a rule is, could be or has been applied, and does not generally refer to material surrounding the element (/ form / word) to which a rule is (/ could be / has been) applied.

not have been necessary if he had not assumed acyclicity. The reference is to *Aṣṭādhyāyī* 6.4.76 *irayo re*: '*re* replaces *ire* (in various Vedic contexts)'. Although ambiguous due to sandhi, the tradition takes *irayo* to be genitive dual rather than genitive singular, which means the dual number has to be explained. The meaning must then be something like '*re* replaces *ire* twice'. The dual here is therefore understood to exceptionally allow this replacement to occur twice to the same element, i.e. the rule is allowed to rewrite its own output once. This is because the substitution *-ire* > *-re* sometimes occurs before roots where there will be secondary insertion of *-i* after the root, resulting in *-i-ire* > *-i-re*; the correct output in such cases, however, is simply *-re*, with secondarily inserted *i* also replaced, so we need to permit this rule to reapply to its own output. Since Pāṇini uses the dual here to license a single reapplication of a rule to its own output, we can infer that Pāṇini presupposes acyclicity.[24]

Altogether, then, it is clear from the phrasing of the paribhāṣā itself, and from the illustrations in the commentary, that this interpretative principle is what modern linguists call acyclicity; the supposed constraint on reuse of contexts is a phantom.

6 Conclusion

I have shown, contrary to Penn and Kiparsky (2012), that Pāṇini assumed acyclicity as a fundamental principle of his grammatical formalism; this can be inferred from the rules of his grammar themselves, and was also recognized and formulated as a principle by the ancient Indian grammatical tradition. As recognized by modern phonology, acyclicity is a fundamentally important constraint on rule systems, restricting the generative power of context-sensitive grammars to that of regular grammars and, correspondingly, making context-sensitive systems computationally tractable. It is a remarkable fact about the ancient grammarian Pāṇini that he anticipated this so precisely.

24 For technical reasons which go beyond the scope of this paper, later commentators such as Nāgeśa and Vaidyanātha do not take *irayo re* to be the authority for the principle of single rule application (which I interpret as acyclicity); instead, they derive the authority for this principle by a series of technical arguments related to the need to avoid multiple application of 8.4.47 (12); see e.g. Kielhorn (1874: 502).

Transliteration

The Sanskrit data is Romanized following the International Alphabet of Sanskrit Transliteration (IAST):
https://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration

Acknowledgements

I am very grateful to Jim Benson, Rishi Rajpopat, and John Coleman, for assistance and discussion of the issues addressed in this paper, and to Martin Everaert for his review of the paper. All errors are my own. The writing of this paper has benefitted from support from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 851990 'LINGUINDIC').

References

Abhyankar, K. V. (ed.). 1967. *Paribhāṣāsaṃgraha: A collection of original works on vyākaraṇa paribhāṣās*. Poona: Bhandarkar Oriental Research Institute.

Backus, John W. 1959. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference. In *Proceedings of the International Conference on Information Processing*, 125–132. UNESCO.

Backus, John W., F. L. Bauer, J. Green, C. Katz, J. McCarthy, P. Naur, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois & J. H. Wegstein. 1963. Revised report on the algorithmic language Algol 60. *The Computer Journal* 5 (4). 349–367.

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

Chomsky, Noam. 1957. *Syntactic structures*. The Hague: Mouton.

Chomsky, Noam. 1959. On certain formal properties of grammars. *Information and Control* 2. 137–167.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

Clark, Alexander. 2015. An introduction to multiple context free grammars for linguists. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.714.8708.

Coleman, John S. 1991. *Phonological representations – their names, forms and powers*. York: University of York dissertation.

Coleman, John S. 1995. Declarative Lexical Phonology. In Jacques Durand & Francis Katamba (eds.), *Frontiers of Phonology: atoms, structures, derivations*, 333–383. London: Longman.

Coward, Harold G. & K. Kunjunni Raja. 1990. *The philosophy of the grammarians*, volume 5 of *Encyclopedia of Indian Philosophies*. Delhi: Motilal Banarsidass.

Culy, Christopher. 1985. The complexity of the vocabulary of Bambara. *Linguistics and Philosophy* 8. 345–351.
Gazdar, Gerald, Ewan H. Klein, Geoffrey K. Pullum & Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Oxford: Blackwell.
Huybregts, M. A. C. 1984. The weak adequacy of context-free phrase structure grammar. In Ger J. de Haan, Mieke Trommelen & Wim Zonneveld (eds.), *Van Periferie naar Kern*, 81–99. Dordrecht: Foris
Hyman, Malcolm. 2007. From Pāṇinian Sandhi to finite State Calculus. In Gérard Huet & Amba Kulkarni (eds.), *First International Sanskrit Computational Linguistics Symposium, Oct 2007,* 13-21. Rocquencourt, France: INRIA. http://hal.inria.fr/SANSKRIT/fr/.
Ingerman, Peter Zilahy. 1967. "Pāṇini-Backus Form" suggested. *Communications of the ACM* 10 (3). 137.
Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. The Hague: Mouton.
Joshi, Aravind K., Leon S. Levy & Masako Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and System Science* 10 (2). 136–163.
Joshi, S. D. & Paul Kiparsky. 1979. Siddha and asiddha in Pāṇinian phonology. In Daniel A. Dinnsen (ed.), *Current Approaches to Phonological Theory*, 223–250. Bloomington: Indiana University Press.
Joshi, S. D. & Paul Kiparsky. 2006. The extended siddha-principle. *Annals of the Bhandarkar Oriental Research Institute* 2005. 1–26.
Kaplan, Ronald M. & Martin Kay. 1994. Regular Models of Phonological Rule Systems. *Computational Linguistics* 20 (3). 331–378.
Kielhorn, Franz. 1874. *The Paribhāṣenduśekhara of Nāgojībhaṭṭa, edited and explained.* Bombay: Government Central Book Depot.
Müller, Stefan. 2016. *Grammatical theory: From transformational grammar to constraint-based approaches*. Berlin: Language Science Press.
Penn, Gerald & Paul Kiparsky. 2012. On Pāṇini and the Generative Capacity of Contextualized Replacement Systems. *COLING* 2012. 943–950.
Pollard, Carl J. 1996. The nature of constraint-based grammar. Paper presented at the Pacific Asia Conference on Language, Information, and Computation, Kyung Hee University, Seoul, Korea. http://lingo.stanford.edu/sag/L221a/pollard-96.txt.
Pullum, Geoffrey K. & Gerald Gazdar. 1982. Natural languages and context free languages. *Linguistics and Philosophy* 4. 471–504.
Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii & Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88 (2). 191–229.
Sharma, Rama Nath. 2001. *The Aṣṭādhyāyī of Pāṇini*, volume V. New Delhi: Munshiram Manoharlal.
Shieber, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8. 333–343.
Staal, Johan Fritz. 1965. Context sensitive rules in Pāṇini. *Foundations of Language* 1. 63–72.
Staal, Johan Fritz. 1966. Pāṇini tested by Fowler's Automaton. *Journal of the American Oriental Society* 86 (2). 206–209.