Pāṇini grammar is the earliest known computing language

John Kadvany[1]

Abstract. Pāṇini's fourth (?) century BCE Sanskrit grammar uses rewrite rules guided by an explicit and formal metalanguage. The metalanguage makes extensive use of auxiliary markers, in the form of Sanskrit phonemes, to control grammatical derivations. The method of auxiliary markers was rediscovered by Emil Post in the 1920s and shown capable of representing universal computation. The same potential computational strength of Pāṇini's metalanguage follows as a consequence. Pāṇini's formal achievement is philosophically distinctive as his grammar is constructed as an extension of spoken Sanskrit, in contrast to the implicit inscription of contemporary formalisms.

1 GRAMMAR AND COMPUTATION

For purposes of this paper, 'computing language' means a formal calculus capable of representing universal computation according to the rules of some formal language whose rules are explicitly described through a metalanguage. In this sense, modern machine and high-level programming languages, by virtue of their formal (meta-)language rules, are computing languages. So too are the classical models of Post, Turing, Church, Kleene and others, including Gödel's formalization of metamathematics as number theory. Though not 'programming' languages intended for machine implementation, the classical models all succeed by virtue of defining 'effective procedure' through a procedure-level formalism which can be then used to represent all such procedures. Frege's first-order logic (as streamlined by Hilbert and Ackermann) may be included here just because, as recognized by Church and Turing, Gödel's number theoretic coding may be translated into the language of first-order logic (and so showing the valid sentences of first-order logic to be undecidable). We tend to think of such formal systems, capable of expressing arbitrary algorithms, as thoroughly modern, certainly as at least late 19th century creations. It's also a modern idea to see how to describe the derivational rules of a formal language *also* through the language, so that object- and metalanguage are one.

But the 19th and early 20th century formalisms for algorithmic expression are not the earliest such, by about two millennia. The first computing language – again, for our purposes, a generic formalism, described through a metalanguage for representing exact generative symbolic procedures of any kind – was devised circa 350 BCE by the Indian grammarian and linguist Pāṇini. The formalism is not identical with Pāṇini's Sanskrit grammar, but is a significant part of it, constituting the grammar's formidable formal methods. Those formal methods are allied with perhaps some centuries of Indian linguistic theory to define the grammar as a whole.

Pāṇini, as put by the late Frits Staal, is the 'Indian Euclid' [1]. Parallel to Euclid's codification of the earliest, if informal, deductive systems, including proof by contradiction, Pāṇini in his Sanskrit grammar formulated and applied the world's first formal metalanguage for generic symbolic manipulation. Pāṇini's basic method was rediscovered in the 1920s and 1930s by Emil Post through his production/rewrite systems. Post *proved* [2], as Pāṇini could not even conceive, that his systems were capable of universal computation. But then that fact has also to be true of Pāṇini's grammar, even as the latter is meant for computationally modest linguistic derivations and not calculation nor computation generally.

To a first approximation, Pāṇini's formal methods, exclusive of his linguistics, are Post's, or vice versa. Pāṇini's grammar has the further remarkable property, relevant to contemporary debates in the philosophy of programming languages, that it is formulated for *oral* recitation, not inscription; indeed, Pāṇini's formalism can be construed as a grammatical generalization of the spoken Sanskrit object language which the grammar describes. In this way, Pāṇini's grammar is in effect the realization of a computing environment as formally recited human speech.

This paper provides a sketch of Pāṇini's grammar and its metalinguistic technology. By way of historical context, the grammar is motivated to construct 'certificates of authenticity', so to speak, for Sanskrit expressions, for both scientific and ideological reasons. Procedural exactness has deep roots in the habitus of Hindu culture, particularly through older traditions of ritual theory, through which the earliest Indian linguistic theories were conceived, including the characterization of grammar as representing continuous speech (*saṃhitā*) using artificial discrete simplifications (*pada,* [3]). As noted repeatedly by Staal, language and linguistics had a preeminent scientific role in ancient India, comparable to geometry and astronomy in Greece, but with a complementary prestige associated in India with algorithmic thinking of all kinds. The oldest theoretical formulations of the topic appear to be those of various so-called ritual 'manuals', guiding explicit ritual design and execution in the *Vedas* and elsewhere. Such were the procedural programming manuals of the time, so to speak.

2 PĀṆINI GRAMMAR

Pāṇini's grammar has long been recognized by linguists ([4], [5], [6]) as the first generative grammar in the modern sense, and capturing Wilhelm von Humboldt's insight that natural language productivity expresses the potentially infinite use of finite

[1] Policy & Decision Science, Menlo Park, California. Email: john@johnkadvany.com

means, an insight also expressed by the Indian grammarian Patañjali (ca. 2nd century BCE, [3]).

Pāṇini's grammar is organized as are many modern grammars, and perhaps all formal languages, as a tiered hierarchy of progressively more powerful representations: the levels roughly being: sounds to phonemes; phonemes to morphemes; and morphemes to syntactically well-formed words and sentences. The grammar includes a great deal of implicit semantics through its linguistic content and a set of basic semantical categories used to initiate derivations, as explained below.

Pāṇini's goal is to describe the (potentially infinite) spoken Sanskrit of his time utilizing the generality of a generative system. Hence Pāṇini has finite sets of what can be thought of as basic symbols which are: the basic set of Sanskrit phonemes (*Śivasūtras*); Sanskrit verbal roots and nominal stems, from which words and then the all-important Sanskrit compound words are formed; and, for metalinguistic purposes, *additional* auxiliary phonemic markers, used as affixes, to control the derivation of Sanskrit words and sentences. 'Derivation' can be taken not entirely, but very much, in the modern sense, as rule-governed, stepwise expression formation. The typical action or event is to rewrite – or 'respeak' – a current expression E with some modified E'.

The user of the grammar, like the user of a formal proof system or programming language, will start with some Sanskrit target word, compound word or sentence in mind as the goal. The grammar is used constructively, like a spreadsheet, and is *not* generally capable of directly 'testing' candidate expressions for grammaticality, though invalid derivations will at some point fail. Needed roots and stems are 'user-selected' to initiate the derivational process, and because Sanskrit is mostly a free-order language, like Latin or Old English, the ordering of these elements is largely irrelevant (through ordering *within* several types of compounds can matter).

From this starting point, metalinguistic rules (*paribhāṣās* – a term created by the tradition but not used by Pāṇini) are used to mark roots and stems as having their intended syntactic roles – using a few simple functional categories which today's linguists may characterize as 'agent', 'goal', 'patient', 'instrument', 'location', 'source' etc. While these choices are 'user-generated', a set of metarules, the *kāraka* rules, list the categories and rules for using them. As put by Paul Kiparsky, "Pāṇini's grammar represents a sentence as a little drama consisting of an action with different participants, which are classified into role types call *kārakas* [which are] roles, or functions assigned to nominal expressions in relation to a verbal root" [7].

From this starting point of the *kāraka* roles and selected proto-words, Pāṇini's metalanguage guides the arduous process of identifying and applying relevant operational rules (*vidhi*) which step-wise transform roots and stems into valid Sanskrit words and sentences, primarily through affixing and compounding. The proper prioritization, exception-allowing, rule-blocking and other use of the operational rules is also laid out by the guiding metarules. The process is comparable to the formation of an individual, concrete and well-formed program by the rules of the programming language in which it is expressed. The 'output' then is a single well-formed word or sentence. All through the process, rule application relies on considerable expertise, and some subjective judgment, for rule identification and application. While employing a rigorous formalism throughout, the organization of rules and their application is subtle and intricate, as happens with the linguistic analysis of many natural languages.

However, given that caveat, rule formulation and application themselves use the modern concepts. First is that of step-wise derivations, as noted. Then, most importantly, rules are codified using what we today think of as 'auxiliary markers' (or 'non-terminals'), simply additional metalinguistic signs whose role is to control the derivational process: what to do if a stem is marked as a past tense verb, what to do if a noun is marked as an instrumental object, how to indicate passive versus active, what sound adjustments to make for adjacent phonemes, and so forth. These auxilary markers, called *IT*, are almost always appended to intermediate strings as affixes (i.e. as *string^affix*) and retained as long as needed, or until the marker is changed or deleted in the derivation. The term *IT* derives from the Sanskrit particle *iti*, used as a quotation marker, and whose deictic status is reflected in allied terms such as *idam/this, iha/here, idānīm/now* [8]. A derivation concludes with application of many phonological rules which effectively convert an expression so that it is ready for speech, particularly by use of *sandhi* rules for adjusting adjacent sound forms. As mentioned above, Indian linguistics long recognized the discrete terms used in their analysis as abstractions; hence derived expressions required 'smoothing' to better approximate empirical speech. The last auxiliary markers for a set of derived words may be deleted, resulting in the finished Sanskrit sentence (case endings and inflections basically dictate sentence structure), akin to a proved theorem or executed computer program. Alternatively, a set of final markers may be retained so that, among other uses, words may be recursively used as components in one of many complex, and compact, Sanskrit compounds, and whose construction is a major focus of all Sanskrit linguistics, not only Pāṇini's.

Here is a sketch of a sentence derivation [9, 10]. Suppose you want to derive a Sanskrit version of "Devadatta is cooking rice in a pot with firewood for Yajñadatta": *devadatta odanam yajñadattāya sthālyām kāṣṭhaiḥ pacati*. The *kāraka* roles chosen would be the verbal *action* of cooking, an *agent* Devadatta, a *patient* of the action which is rice, an *instrument* of firewood, a *location* of the pot, and a *recipient* Yajñadatta of the action. The *kāraka* categories are formally defined and regulated by metarules, and provide a powerful heuristic for constructing a wide range of sentences. The categories mediate informal semantic meaning through their functional syntactic role. The free word-order of Sanskrit means the selection initial stems, roots or words to associate with *kāraka* roles can be thought of as an unordered set: {*devadatta, firewood, rice, cooking, pot, yajñadatta*}, i.e. {*devadatta, kāsṭha, odana, pac, sthāli, yajñadatta*}. So again, Kiparsky: the grammar is a "pure form of lexicalism."

These elements now require rule applications to mark their assigned *kāraka* roles and to create new expressions. For example, the pot is singular and the location of the action, and that is marked by the suffix –**ṅ***i*, producing *sthāli*-**ṅ***i*. The bold face **ṅ** represents an auxiliary and non-terminal marker used in the derivation process, with italicized *i* being a terminal sound, and the hyphen indicating concatenation. Similarly, *yajñadatta* is the recipient of the action, marked by the suffix –**ṅ***e* and yielding *yajñadatta*–**ṅ***e*. The patient and instrumental roles, for rice and firewood respectively, can be marked with suffixes not needing auxiliary markers: *odana-am* and *kāsṭha-bhis.*

Derivation of the verb and its inflection for the cooking action, *pac*, involves more steps. There is first an assignment of the present tense using the suffix –**laṭ**, chosen from a set of **l** suffixes (*lakāra*) including perfect, imperfect, subjunctive, imperative, and other tenses. The verb can also refer actively to the agent Devadatta (*cooking the rice*…), or passively to Devadatta by focusing on the rice (…*cooked by Devadatta*), an example of non-deterministic choice in the derivation. Devadatta is singular and is cooking for another, leading to the –**laṭ** suffix being replaced by –*ti***p**. The verb root *pac* also happens to require the vowel *a* between root and suffix, leading to *pac*-**ś***a***p**-*ti***p**. To now consistently mark the agent Devadatta as actively cooking, as planned with the active verb and required by the marker ti**p**, means use of the –s**u** suffix on *devadatta.* The marked-up roles lead to {*devadatta*–s**u**, *kāsṭha-bhis, odana-am, pac-* **ś***a***p**-*ti***p**, *sthāli*–**ṅ***i, yajñadatta*–**ṅ***e*}. An important caveat: each 'step' involves several substeps to identify the operational rule which can actually be applied. Those substeps may involve numerous cross-references in the grammar or the application of metarules to resolve rule conflicts, possibly extending across several of the grammar's eight 'books'. Pāṇini's complex derivations in this way differ significantly from those found in most modern formalisms.

Given that, the intermediate expressions can be used to derive actual words by deletion of non-terminal markers **u**, **ś**, **p**, **ṅ,** and derivation of correct terminal sounds through phonological rules. The derivation is typical in that auxiliary markers are similarly used throughout the grammar for rule expression and their application. Sanskrit syntax is already highly governed by case endings, so this is the basic means by which *Pāṇini's grammar extends the object language by its, the object language's, own means*. The systematic role for affixing makes Pāṇini's innovation a kind of (meta-) grammaticalization, a linguistic transformation which is often central to language change generally [11]. In modern computational theory, the analogous bootstrapping innovation will be to use rewrite rules to formulate a metarule for all rewrite rules; or to use Turing machine grammar to define a universal Turing machine; or as shown by Gödel, to use number theory to define a metalanguage for its own derivations; and so forth. The bootstrapping principle also occurs practically when a programming language like C or Pascal is used to write its own compiler, with successive versions accommodating larger swaths of the language. Here, though, what differs is that we assume the spoken natural language Sanskrit and its structure to begin with. The object language is neither a mathematical invention nor is it written, at least in principle.

3 PĀṆINIAN COMPUTATION

The basic claim then is that Pāṇini's system is sufficiently detailed to qualify as the earliest known computing language. Here are supporting details, along with several caveats. The idea of metalanguage and object language is, remarkably, fully understood by Pāṇini, indeed it is the modus operandi for his approach. The *paribhāṣā* metarules elaborate *how* the system is to be used, which is to apply operational rules to increasingly transformed symbolic expressions. As stated, and as illustrated by the example, the principle method to express rules and rule application is through the auxiliary markers.

That central role for auxiliary markers means, quite simply, the method of rewrite systems made famous by Emil Post starting in the 1920s but not recognized through publication until the 1930s [12]. The method is very much here, with even more formal rigor than found in Euclid's derivations.

Pāṇini goes so far as to devise a quite general formulation of a method he applies repeatedly, especially in his phonological rules, that of *context-sensitive* rules. We express those today as, e.g., $A \rightarrow B / C\_\_D$, meaning: replace expression A by B when A just follows C and D just follows A, with C or D possibly empty – so deletions can be treated as a kind of replacement. Pāṇini formulates an equivalent notion of generic string positions and their roles, perhaps his most elaborate formal construct which is directly comparable to a modern equivalent. So it's not possible to argue that Pāṇini has some serendipitous notion of rewrite rules; to the contrary, he developed the first expression of one of the central ideas of the modern theory. The theory of context-sensitive and related rule types was initiated by Chomsky and others in the 1950s and building on Post's ideas.

Here is an illustration of how Pāṇini's formal terminology works for a phonological rule [1]. The rule is to replace *i* by *y* when followed by any of nine Sanskrit vowels. That could be expressed as nine separate rules, but is better codified by a single rule referring to a right-context *D* of "all following vowels" (and null left-context *C*). Similar replacements *u*→*w*, *ṛ* →*r*, *ḷ* →*l* occur, again with any following vowel. In modern terms, this means a summary rule to be codified is the *ordered* replacement <*i, u, ṛ, ḷ* > → <*y, v, r, l*> when followed by a vowel, meaning a phoneme from the list {*a, i, u, r, l, e, o, ai, au*}. This list and others are coded as sublists in the *Śivasūtra* phoneme set, interpreted as being ordered as fourteen separate "rows". Sublists of phonemes are identified by auxiliary markers for "start" and "endpoints", with those markers skipped or deleted in the sublist enumeration; that guidance is also spelled out as a metarule. It's also worth noting that Pāṇini's phoneme set, the *Śivasūtras* (suggesting deliverance by the god *Śiva*), while nominally expressed as a sequence of linear *sūtras* is apparently optimally designed to enable its systematic reference to some 42 sublists of vowels, consonants, etc. [13]

So, in the *Śivasūtras, i***k** stands for {*i, u, ṛ, ḷ* } and a**c** refers to {*a, i, u, ṛ , ḷ , e, o, ai, au*}, which is the vowel list needed above; the braces {…} are our written convention. The other list needed is *ya***ṇ** or {*y, w, r, l*}. A *sūtra* applies for taking same-sized pairs of lists as ordered sequences instead of unordered sets; the rule basically allows the definition of finite mappings between defined lists.

Given names for desired lists (e.g. *i***k**, *a***c**, *ya***ṇ**), the second step is using them to construct a context-sensitive rule $A \rightarrow B / C$ $\_\_ D$. The challenge then is to define these functional roles for *A, B, C, D*. Pāṇini's solution is to give the lists, through their names (*i***k**, *a***c**, *ya***ṇ**), *kāraka* case endings in a *sūtra* statement, and thereby to grammaticalize the rule. That is, the case endings are added to the names of the lists, treated as syntactic objects, to contextually define their roles in stating a context-sensitive rule.

Pāṇini's artificial case endings are therefore used to express "in the place of *A*, substitute *B*, when after *C* and *D* follows", using several metarules: *genitive* case ending marks *A* as what is to be substituted; *nominative* case ending marks *B* to substitute for *A*; *ablative* case ending marks a preceding segment *C*;

locative case ending marks a trailing segment *D*. The context-sensitive conventions also are used as a master format for *sūtra* coding and hence are a consistent clue to their meaning, with many operational rules framed in *sūtra* form as $A_{Genitive}$ $B_{Nominative}$ $C_{Ablative}$ $D_{Locative}$. [7]. In the (well-known) example, the rule leads to: *i***k** + genitive, *ya***ṇ** + nominative, *a***c** + locative, or {*ikaḥ*, *yaṇ*, *aci*}. When the words are combined in that order, a sound-changing *sandhi* rule completes the derivation as *iko yaṇ aci.* The rule in effect is a metalinguistic sentence which is meaningless in Sanskrit proper. That is a remarkable use of Sanskrit to bootstrap itself into a metalanguage. The construction is possible because of our ability to consider the Sanskrit object language as 'data' subject to grammatical rules. The expression of those rules as an extension of Sanskrit speech is a profound illustration of the role of intentionality in language use, grammar formulation and, by implication, computation.

The clarity and directness of Pāṇini's system also comes at considerable cost. There are thousands of rules and metarules, organized as eight 'books' (*Astadādhyāyī*). The dependencies across rules, and the organization of rules into subgroups controlled by marked 'headings', are highly complex. The system should be thought of as *containing* a rigorous rewrite formalism, especially through the metalanguage, but with the grammar as a whole organized using many intricate linking, structural and referential devices making Pāṇini's system *sui generis* [9]. Critically, rules are codified as some 4,000 brief and memorizable *sūtras*, sometimes wrongly identified with the grammatical rules themselves.

These mnemonic expressions are decoded and recoded through the ongoing oral tradition of grammarians, who have evolved nomenclature and guidelines for stating Pāṇini's rules in explicit form, along with examples, variant interpretations and criticism. Whether Pāṇini's grammar was originally *formulated* without inscriptional aids is unknown, certainly controversial, and quite doubtful for some [14]. However, even assuming considerable inscriptional help, the finished product is highly refined and ready for oral expression by communities of experts. A comparison is possible to the iterated construction of early machine and programming languages through their expert communities of esoteric practice, with the concise formalization of their work following as a final product. With little exaggeration, Pāṇini's grammar itself, both in complexity and organization, is like a user's manual for an early operating system *plus* a programming language running under it – with both expressed using a single formalism *and* using speech as its phonemic 'hardware'. The *Śivasūtras* are indeed organized in terms of place and type of articulation in the vocal apparatus, and so realizes language as physiological mechanism.

Let's return to our main point, regarding the implicit computing power of Pāṇini's system. In terms of the techniques Pāṇini needs and explicitly uses for his linguistic theory, the 'maximum' is that of context-sensitive rules. These rules are already, in the view of some modern linguists [15, 16] overkill for natural language syntax (compared to a context-free syntax), which largely is Pāṇini's scope too. But it is a simple observation, given the explicitness of Pāṇini's metalanguage, that his grammar can be directly *extended*, using his *same* methods of auxiliary markers, to represent any rewrite system desired [17]. Emil Post's achievement was to show that it was just the method of auxiliary markers which could be used to simulate the derivations of any rewrite system. That is how Post demonstrates that his production/rewrite systems are equivalent to the representational power of Turing machines. All the heavy lifting to define the metalinguistic framework is completed by Pāṇini. He has just *limited* his target application to be the grammatical expressions of spoken Sanskrit. For that, he needs a complex linguistic theory, and a precise metalanguage for codifying the grammar of his Sanskrit object language. So the computing power *needed* by Pāṇini is *not* 'universal', but he has put everything in place for just such a computing environment.

4 MEDIA

For Pāṇini that environment was neither a computing device nor even the inscriptions constituting a proof. The grammar was meant to be used as oral *recitation*, notwithstanding our contemporary written compilations of the *Aṣṭādhyāyī* (e.g. [9]). Genuine users of the grammar, which I am not, really should learn it through oral training and practice with an older generation, starting preferably in one's youth to better internalize correct pronunciation and articulation of object and metalanguage. Practically, the oral medium makes Sanskrit derivation hard enough, though it's been said that the recitation of the whole grammar, at least in *sūtra* form, can be completed in some several hours. So, realistically, extending the grammar to an oral calculus, as it were, to computations involving positionally represented numbers and multiplication – a *sine qua non* for all computation – is at most a thought-experiment in 'Pāṇini arithmetic' – yet not so different from Turing's thought-experiment involving symbolic inscription. The thought-experiment is just to mimic Post's universal rewrite system in Pāṇini's rewrite system, almost by direct translation using some few new symbols for 'numbers', 'axioms', 'proofs' and other needed categories and operations.

Here is the linguistic and philosophical point to that exercise. Because Pāṇini's system is explicitly designed as an *extension* of his Sanskrit object language, we therefore have an example of universal computation formulated as an extension of a natural language by its own means. As said above, at a detailed level this is accomplished via a kind of grammaticalization, but carried out consciously and purposively, unlike the same process occuring in historical language change. The great twentieth century logicians all had to devise their own – logical and mathematical – versions of the later idea that 'programs are data'. That includes Gödel's number-theoretic coding of number theory proofs; Post's canonical rewrite rules; Turing's universal machine; Church's lambda expressions; and so on. The key step is always to show that a particular formal language can be used as its own metalanguage – with each such formulation having its own theoretical importance and claim to fame. Pāṇini precedes exactly Post in method, of course without Post's *proofs* of comparative computing power, but amazingly, *as* a natural language grammar formulated using the same grammatical devices as the target language, principally affixing and inflectional changes. Panini saw how the intrinsic resources of the spoken Sanskrit of his time could themselves be used to formulate a metalanguage for exact description of Sanskrit as the object language.

In this way, Pāṇini created the first computing language and in oral form. In contrast, Turing explicitly uses a thought experiment involving symbolic inscription, with that media is implicit in all other modern formalisms and their metaphors (e.g. Church's lambda calculus is also a rewrite system). The reason oral codification was important for Indian linguists is that Sanskrit *speech* was 'the language of gods in the world of men' [18], with writing even potentially polluting, and Sanskrit also being the language of ancient Indian science, useful for transmitting ritual and astronomical knowledge accurately over a huge land mass.

From a modern perspective, while inscription has undoubted benefits in objectifying and memorializing natural and artificial languages, there is a received dogma that computation can be expressed in any media you like [19, 20], with software ultimately an abstraction independent of any hardware implementation. We therefore now have now a real historical example of just that media freedom, but in human speech, which along with the gestures of signing, is a primal expressive media, of natural language, at least for us modern humans [21]. Philosophically, Pāṇini's example shows that the differences between natural and artificial computing languages are much smaller than often thought. Not because natural languages are, or are close to being, computing languages, but because the construction of computing languages is apparently just a continuation of natural language constructions by their own means [22].

REFERENCES

[1] F. Staal, F. *Universals*. Chicago: University of Chicago Press (1988).

[2] M. Minsky. *Computation: Finite and Infinite Machines*. New York: Prentice-Hall (1967).

[3] F. Staal. *Discovering the Vedas: Origins, Mantras, Rituals, Insights.* Kundli, India: Penguin (2008).

[4] L. Bloomfield. *Language*. London: George Allen and Unwin (1933).

[5] N. Chomsky. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press (1965).

[6] N. Ostler. *Empires of the Word*. New York: Harper Collins (2005).

[7] P. Kiparsky. Pāṇinian linguistics. In *Encyclopedia of Languages and Linguistics* 2nd ed. K. Brown et al. eds. New York: Elsevier Science (2005).

[8] F. Staal. The Concept of Metalanguage and its Indian Background. *Journal of Indian Philosophy* **3**: 315-354 (1975).

[9] R. N. Sharma. *The Aṣṭādhyāyī of Pāṇini* I (six volumes): *Introduction to the Aṣṭādhyāyī as a Grammatical Device*. New Delhi: Munshiram Manoharlal (1987) .

[10] B. Gillon. Pāṇini's *Aṣṭādhyāyī* and Linguistic Theory. *Journal of Indian Philosophy* 35:445-468 (2007).

[11] Hopper, P. and E. Traugott. *Grammaticalization* (2nd ed.). New York: Cambridge University Press (2003).

[12] A. Urquhart. Emil Post. In Gabbay, D.M., Woods, J. (eds.) *Handbook of the History of Logic, Volume 5: Logic from Russell to Church*. Amsterdam: North-Holland (2009).

[13] W. Petersen. A mathematical analysis of Pāṇini's Śivasūtras, *Journal of Logic, Language, and Information* **13**: 471-489 (2004).

[14] J. Goody. *The Interface Between the Written and the Oral*. New York: Cambridge University Press (1987).

[15] P. Culicover. Review of R. Huddleston and G. Pullum (eds.) *The Cambridge English Grammar*. *Language* **80**: 127-141 (2004).

[16] G. Pullum. and B. Scholz. Contrasting applications of logic in natural language syntactic description. *Logic, Methodology and Philosophy of Science: Proceedings of the Twelfth International Congress*. ed. Petr Hájek et al. 481-503 London: King's College Publications (2005).

[17] J. Kadvany. Positional value and linguistic recursion. *Journal of Indian Philosophy* **35**: 487-520 (2007).

[18] S. Pollack. *The Language of the Gods in the World of Men: Sanskrit, Culture, and Power in Premodern India.* Berkeley: University of California Press (2006).

[19] B. J. Copeland. What is computation? *Synthese* **108**: 335-359 (1996)

[20] J. Searle, J. *Philosophy in a New Century: Selected Essays*. New York: Cambridge University Press (2008).

[21] M. Tomasello. *Origins of Human Communication*. Cambridge, MA: MIT Press (2008).

[22] M. Tomasello. *Constructing A Language: A Usage-Based Theory of Language Acquisition*. Cambridge, MA: Harvard University Press (2003).