Formal Sanskrit Syntax: A Specification for Programming Language

K. Kabi Khanganba, Girish Nath Jha
School of Sanskrit and Indic Studies, Jawaharlal Nehru University, New Delhi, India
khanganba.360@gmail.com, girishjha@gmail.com

Abstract

The paper discusses the syntax of the primary statements of the *Sanskritam*, a programming language specification based on natural Sanskrit under a doctoral thesis. By a statement, we mean a syntactic unit regardless of its computational operations of variable declarations, program executions or evaluations of Boolean expressions etc. We have selected six common primary statements of declaration, assignment, inline initialization, *if-then-else*, *for* loop and *while* loop. The specification partly overlaps the ideas of natural language programming, Controlled Natural Language (Kunh, 2013), and Natural Language subset. The practice and application of structured natural language set in a discourse are deeply rooted in the theoretical text tradition of Sanskrit, like the sūtra-based disciplines and Navya-Nyāya (NN) formal language, etc. The effort is a kind of continuation and application of such traditions and their techniques in the modern field of Sanskrit NLP.

1 Introduction

The paper is based on a non-English-based programming language, called *Sanskritam* being developed under a doctoral thesis. Instead of adopting the syntax of programming languages highly inspired by the symbolic language of logic and mathematics, we have adopted a natural language-based syntax like those of generic Sanskrit used to write the Aṣṭādhyāyī [1] (AD), a grammar of the natural Sanskrit itself. We have selected 6 common primary statements that any general programming language may have. By a statement, we refer to a syntactic unit regardless of its computational operations of variable declarations, program executions or evaluations of Boolean expressions etc.

The program statements are translated to the two basic types of natural language sentences of assertive and imperative in the present tense, active voice. One of the core components in a program expression is the 'operator', which contributes to the semantic decision. Different keywords and signs are used to indicate the operation of a statement in semantic analysis. In the whole paper, *Sanskritam* refers to the language being specified while Sanskrit means the very natural *Sanskrit*.

India has had one of the oldest traditions of linguistics, philosophy, logic, mathematics, and many other disciplines which were almost all written in Sanskrit. The language of Western symbolic logic is successfully adopted by a number of the modern field of studies, including mathematics, computer science, computer programming languages, formal language grammars, physical science and biology (Bhattacharya, 1990). The Indic logic and other disciplines follow a linguistic model for their systems instead of a symbolic model. The technique or logic of such a linguistic model, be it Sūtra [2] or NN (an Indic philosophical school) language, had successfully gained popularity and share among other Indic fields of studies also (Bhattacharya, 2006). They are widely accepted, used, and taught in school/universities, and there is a good number of active communities. The artificial formal language known as Navya-Nāya-

[1] A 5th BCE Sanskrit grammar written in a semi-formal Sanskrit itself by Panini.
[2] A kind of logically arranged text that deals a discipline or science. Literally it means a string or thread; an aphorism. Sometimes it again means an individual Sūtra many of which constitute the whole Sūtra text.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 72–78

Bhāṣā or NN language is adopted by other Indic disciplines to define their domain-specific terms. On the other hand, there is no wonder about the Sūtra technique of learning and teaching Sanskrit even today among the Sanskrit community. However, despite its relevance, usage and living community in the present, the Indic linguistic model of logic tool lacks adapting to the modern technical disciplines, unlike the evolution of western symbolic language into electronic machine languages.

Sanskrit computational linguistics (SCL) has an active international research community (Jha, 2010). Apart from this and more importantly, SCL is taught as a post-graduate course or subject at many Sanskrit departments in Indian universities and IITs. There are more than seventeen Sanskrit universities including three central Sanskrit universities and hundreds of Sanskrit departments in many central and state universities in the country where SCL can be introduced and taught. The post-graduate Sanskrit students generally have different non-science background like traditional *gurukula*[3] education system or arts stream. When the students come to master level and opt for SCL, they suddenly learn primary programming languages which use a number of punctuation marks, mathematical notations and English-based keywords etc. Due to non-multidisciplinary and inter-disciplinary mode of education system except an introduction (to multidisciplinary education system) in the National Education Policy 2020, the students are not familiar with symbolic logic which are commonly used in mathematics, computer programming languages and general sciences. However, by pre-master level, Sanskrit students are well introduced to formal linguistic model of logic of traditional Paninian grammar etc. Sanskrit students of different traditional Sanskrit disciplines of *sāhitya* (literature and literary theory), *darśaṇa* (philosophy), *jyotiṣa* (astronomy and calendar) etc. have to study compulsorily and primarily the Panini grammar. They must better adapt a high level programming language based on such traditional formal Sanskrit of *sūtra* or NN artificial language.

2 Related Work

This work is related to formal language specification based on natural language for development of general programming languages. The Inform 7 or Natural Inform is a highly domain specific programming language based on natural English for writers of interactive fiction [4]. However, our work focuses on specification of equivalent natural Sanskrit-based formal statements for common primary statements of general programming languages. Controlled Natural Languages or CNLs are designed by selecting subsets of features like vocabulary, morphology, syntax, semantics and pragmatics from a particular natural language of the designer's choice (Kunh, 2013). CNLs have been proposed for different applications like knowledge representations, rule systems, equerry interfaces and formal specifications. This concept partly overlaps with the *Sanskritam* specification that adapts the primary features of natural Sanskrit like syntax and morphology.

There is no such specification of Sanskrit or Sanskrit-based programming language in which even the keywords are borrowed from Sanskrit. However, two Indic disciplines of Grammar (especially the Paninian School) and NN influence the Sanskritam specification.

Two major aspects of Paninian grammar concerned with our specification are 1) the language of the grammar, a generic natural Sanskrit-based specification mostly formal (Huet, 2016; Kadvany, 2016), and 2) the logic of the grammar. A subset out of the standard classical Sanskrit is used for the AD[5] grammar (Deshpande, 1991). The subset is a kind of bootstrapping used to define the natural Sanskrit itself (Kadvany, 2016). This subset exhibits most of the primary features of the natural Sanskrit syntax, morphology, Sandhi, and Samāsa (compound). It avoids using finite verbs, frequently uses abstract and verbal nouns, and compounds with oblique case relations between their members. The parsing of Sandhi and Samāsa is still in its inception stage in Sanskrit NLP, so we skip this feature for the time being in our specification. The AD is a list of rules called *sūtra*(s), logically arranged and related to each other. Like a program source code, the order of rules and their arrangement matter very much in

[3] Traditional Indian institutional system.
[4] http://inform7.com/about/
[5] AD as the primary text along with other auxiliary texts and lexicons constitute the Paninian grammatical system.

rule interpretation. Out of 7 types [6] of rules (Sharma, 2002), the *sañjñā* (definition) rule defines a term or variable before it is used elsewhere which overlaps the idea of statically typed language. Vidhi (executive) rule does all types of operations or executions. Huet (2016) highlights the possibility to reconsider the AD as a high-level program compiled into some low level machine code.

The NN language is a completely generic natural Sanskrit-based formal language that primarily deals with unambiguously presenting a concept of an object by fully exploiting the abstraction feature of natural Sanskrit and linguistics theories of Paninian system (Bhattacharya, 2006). Since it is based on natural language, a NN statement or sentence may be practically cumbersome and resemble a long version of a symbolic formula. This is more suited for object oriented programming specification which we may apply later on.

3 Formal Sanskrit

In this section, we will discuss the syntax of six basic statements of Sanskritam specification. We selected the primary statements of general programming languages, mostly inspired by Java (Schildt, 2019). We use the IAST [7], a Latin alphabet-based transliteration for Sanskrit texts. To represent the formal syntactic structure, we use the grammar notation of ANTLR metalanguage specification (Parr, 2013). The *Sanskritam* grammar has its terminal symbols and tokens defined by its lexical grammar, representing program keywords, the nominal and verbal roots, indeclinable and other required word classes of the natural Sanskrit.

$$statement : np\ complement\ copula?\ ';'$$
$$\mid complement\ np\ copula?\ ';'\ ;$$

The tokens in lower case not closed by single inverted commas represent non-terminals. The left-hand side of the colon represents a rule name which can have one or more alternate rules separated by bar (|) and terminated by a colon.

A Sanskritam statement is a simple copular sentence that constitutes the NPs (of subject) and complement/s, an optionally omitted copula[8], and finally terminates by a semicolon (;). The copula cannot be in a passive or impersonal [9] construction. Since a complement represents a predefined term or data type, it allows the subject to occur before or after its complement expression. For instance:

SAN: vṛddhih ād aic ;
ENG: "ād (vowel *ā*) aic (diphthongs *āi* and *au*) (are) Vṛddhi."
SAN: ad eṅ guṇaḥ ;
ENG: "ad (vowel *a*) eṅ (diphthongs *e* and *o*) (are) Guṇa."

The predicative expression *Vṛddhi* precedes its subject *ād* (and) *aic* representing program variables in the first statement while the subject *ad* (and) *eṅ* precedes the complement in the second statement. The copula is omitted in both of the statements.

3.1 The Declaration Statement

Declaration statement is optionally a zero copular sentence that consists of a subject NP as the identifier (the name of a variable) and the name of a data type as the subject complement. The copula agrees with the subject NP. The subject NP may include one or more singular number nouns separated by a comma to support declaration of more than one variable of the specified type in a single statement. Dual and plural paradigm of nouns are not presently defined and reserved to represent array variables. The conjunction *ca* meaning *and* can also optionally join two or more variables. The positions of the NP and the complement may be interchanged.

$$declaration : subj\ (',' subj) + complement\ copula?\ ';'\ ;$$

where *subj* is a noun ending with nominative case marker. *asti, staḥ* and *santi* – the three paradigms of the copula '*as*' meaning *to be* are defined as the keywords for the declaration of one, two, and more variables respectively within the same statement.

SAN: chātraḥ al ;
ENG: "chātra (is a) String."
SAN: chātraḥ al asti ;

[6] The rules in AD are divided into 7 categories on the basis of their operations.
[7] https://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration
[8] Sanskrit has two basic copula roots called *as* and *bhū*. They are inflected for tree tenses and seven moods like other common Sanskrit verbal roots.
[9] Sanskrit verb has three inflectional voices: active, passive and impersonal

ENG: “chātra is (a) String.”
SAN: chātraḥ, viṣayaḥ al ;
ENG: “chātra, viṣaya (are) String.”
SAN: chātraḥ, viṣayaḥ al staḥ ;
ENG: “chātra, viṣaya are String.”
SAN: chātraḥ, viṣayaḥ ca al ;
ENG: “chātra and viṣaya (are) String.”
SAN: chātraḥ, viṣayaḥ ca al staḥ ;
ENG: “chātra and viṣaya are String.”

3.2 Assignment Statement: Assigning a Value to a Declared Variable

The assignment statement constitutes two juxtapositional interchangeable NPs: a subject NP (NPS) and an NP of locational complement (NPLC). The imperative paradigms (syāt, syātām, syuḥ) of the copula ‘*as*’ are optionally omitted. The copula occurs at the end of both of the NPs. The noun in NLCP represents the assignee, a declared variable.

assignment : subj nGenetive copImp? ‘;’
| nGenetive subj copImp? ‘;’ ;

nGenetive and *copImp* are a noun ending in genitive and copula in imperative mood respectively.

The assignment statement adapts the metarule *ṣaṣṭhī sthāneyogā* [10] from AD. It defines that *ṣaṣṭhī* or the genitive marker indicates a *sthānī* or a locus, an element that holds another value/element.

SAN: nāma chātrasya ; //or
SAN: chātrasya nāma syāt ; //or
SAN: nāma chātrasya syāt ;
ENG: “nāma replaces chātra.”

The NPLC can optionally have the keyword ‘*sthāne*’, locative paradigm of the ‘*sthāna*’ in singular to express “*in the place”*. For instance, the AD rule

SAN: iko yaṇ aci
ENG: “yan (must be) of ik (if) ac follows” (literal)
ENG: “yan must place ik if ac follows”

It has the alternative expression

SAN: ikaḥ sthāne yaṇ syād[11]
ENG: “yan must be in the place of ik if ac follows.”

The Kāśikāvṛtti, a commentary on AD interprets that *sthāna* is synonymous with ‘*prasaṅga’* meaning ‘*context’*.

SAN: *sthānaśabdaśca prasaṅgavācī*[12]
ENG: *“sthāna means context”*

Thus, the following two alternative constructions are possible:

SAN: chātrasya sthāne nāma ;
ENG: “nāma in the place of chātra.”
SAN: chātrasya sthāne nāma syāt ;
ENG: “nāma must be in the place of chātra.”

A string literal defined within double quotes can take the role the subject NP followed by the direct marker *iti*.

SAN: chātrasya sthāne “dīpakaḥ” iti ;
ENG: “the “dīpaka” must be in the place of chātra.”

This statement allocates a value to a variable. After the declaration of a variable, a value is assigned to it with an assignment statement. In general programming languages the equal sign = is the key token used to indicate the assignment operator.

3.3 Declaration and Assignment: (**Inline initialization statement**)

The inline initialization statement is an expression of both of the declarative and assignment statements within a single sentence. This statement is operationally a compound statement. It consists of two operations: 1. Declaration of a variable, and 2. Assignment of an initial value to that variable. It contains two clauses separated by a comma, including one main clause followed by a relative clause in order. Both of these two clauses are derived from the very two main clauses of the declarative and the assignment statements. The main clause within an inline initialization statement is a declarative statement itself. The relative clause represents an assignment statement introduced by a possessive form of the relative pronoun ‘*tat’* to meet the genitive metarules. The antecedent is provided by the subject NP of the main clause.

inlineInitialisation : assignment rePro subj ‘ca’ cop? ;

The relative pronoun agrees with its antecedent in number and gender, the subject NP within the main clause. In the example below, the relative pronoun *tasya* (of that) establishes an anaphoric relationship with the antecedent *chātraḥ*, which is masculine and singular.

SAN: chātraḥ al asti, tasya nāma ca syāt ;

[10] Sthāna means prasaṅga or context. That which constitutes context is called sthāneyogā. The locative marker in the word sthāneyogā appears exceptionally even after compounded. The un-compounding form of this compound is *sthānena yogo yasyā* meaning that which has the context - prathamāvṛtti, AD 1.1.49.

[11] Siddhāntakaumudī on AD 6.1.77.

[12] Kāśikāvṛtti on AD 1.1.49.

ENG: “chātra is a string, and nāma must replace that.”

3.4 *if-then-else* Statement

The *if-then-else* statement is a complex sentence. It consists of two clauses: 1) the block of *if conditional clause* and 2) the block of *main clause*. The *if-clause* precedes the *main clause*. An *if-clause* expresses and checks one or more conditions prior to the other one or more consequences. The *if-clause* begins with the conditional marking word *yadi* meaning *if*. The *if-clause* contains one or more affirmative or negative or both of the copular sentences to express a boolean condition or conditions. Two or more conditional sentences are separated by a comma and optionally closed by the conjunction *ca* (and) just before the introduction of the following main clause. The *if-clause* is immediately followed by a consequence, the main clause representing consequences. The *tarhi* (then), which is the corresponding consequent marker of the *yadi* (of the *if-clasue*) begins the main clause.

ifThenElse : ifStatement elseIfStat elseStat? ';' ;*
ifStatement : 'yadi' conditions+ 'tarhi' statements ;*
conditions : conditon moreConditions ;*
moreCondition : (',' condition)+ 'ca' ;
elseIfStat : 'yadi' 'ca' conditions+ 'tarhi' statements ;*
elseStat : 'anyaTA' statements ;*

An *if* phrase expresses that one action or situation must happen first before the other one will/can happen. The action in the clause expresses either: (1) a condition for a singular outcome to occur or (2) a recurring situation "whenever" with a predictable outcome (in general).

If the condition/s in the first clause is true, then the statement in the second clause is executed else, the second clause is bypassed. Here is a piece of Sanskritam code to show the if-then-else statements:

SAN: chātraḥ al ; // al=string declaration
ENG: “chātra (is) string.”
SAN: chātrasya "dīpakaḥ" ; // assignment
ENG: “chātra (must hold) “dīpakaḥ”.”
SAN: kramāṅkaḥ saṅkhyā ; // saṅkhyā = int decl.
ENG: “krāmāṅka (is an) int.”
SAN: krāmāṅkasya 1 ; // int = 1;
ENG: “krāmāṅka (must hold) 1.”
//conditional statement --------->
SAN: yadi 1 krāmāṅkam vyapnoti, chātrasya "raviḥ" ;
ENG: “if 1 pervades krāmāṅka, chātra (must hold) “ravi”.”
SAN: yadi 1 krāmāṅkam vyāpnoti tarhi chātrasya "raviḥ";
ENG: “if 1 pervades krāmāṅka then chātra (must hold) “ravi”.”

We have reserved the metarule of locative marker which is specified in AD:

SAN: tasminniti nirdiṣṭe pūrvasya.
ENG: *“*A word in the locative case indicates that an operation must happen on an immediate previous entity.*”*

This rule defines a metarule of a conditional statement that checks the boolean condition of an element following another element. This rule restricts the locative marker to mark a following element. For instance, x-locative checks the boolean condition of x that comes after a preceding element. Here is an example rule from the AD itself:

SAN: iko yaṇ aci.
ENG: “yaṇ replaces ik (if) ac follows (ik).”

The *aci* (*ac*-locative) is declined in locative. The *tasminniti nirdiṣṭe pūrvasya* checks the boolean condition of occurrence of *ac* after *ik.*

3.5 For Loop Statement

For the *for-loop* statement, we adopt the basic syntax of header rule or Adhikāra rule from AD. A header rule in AD is a rule that initiates a scope marking its boundary generally, until a next scope starts. Morphologically, a header rule constitutes the name of the scope declining in locative case like in the form of x-locative. For instance, Kārake means Kāraka-locative meaning in (the scope of) Kāraka (relational dependency). We selected the key word *Āmreḍite*, the locative paradigm of the word *Āmreḍita* meaning repetition (Williams, 1872) to mark *for-loop*. Thus, the *for-loop* statement consists of a header sentence followed by a list of statements and finally closed by the direct marker *iti.*

forStatement : (Amredite | Amreditasya Avasthayam) forInitializer forTerminator Anupurva '-' statBlock Iti Amreditam? ';' | forInitializer forTerminator Anupurva '-' statBlock Iti Amredaya ';' ;

Symbols initialized by upper case are lexical rules representing terminal symbols. Tokens within the single inverted commas are terminal symbols which are to appear in the program exactly as written. Three examples have some slightly different optional keywords are given below:

//for loop : 1st Syntax
SAN: āmreḍite vepakāt āvepitam anupūrvaśaḥ -
ENG: “in āmredita from vepaka to vepita one by one”
…….
“statemets….”
SAN: iti;
ENG: “end;”
//for loop : 2nd Syntax
SAN: vepakāt āvepitam anupūrvaśaḥ -
ENG: “from vepaka to vepita one by one -”
…….
“statements……”
SAN: iti āmreḍaya;
ENG: “thus repetit (them);”
//for loop : 3rd Syntax
SAN: āmreḍitasya avasthāyāṃ vepakāt prākvepitāt anupūrveṇa –
ENG: “in the condition of āmreḍita from vepaka to vepita one by one”
……..
“statements….”
SAN: iti āmreḍitam;
ENG: “thus āmreḍita;”

3.6 While Loop Statement

The *while* loop also consists of two clauses, one subordinate and one main clause respectively representing the two blocks of a common while loop statement like in Java. The subordinate clause represents one or more conditional expressions and main clause can consist of one more statements separated by commas representing the code block to be evaluated.

whileStatement: Yavat conditionStatements Tavat statBlock Iti';' ;

The keyword *Yavat* meaning *while* begins the subordinate clause which is just immediately followed by the keyword *Tavat*, a correlative of the *Yavat* begins the main clause.

SAN: Yāvat vepitaḥ vepakāt nyūnataraḥ Tāvat
ENG: “while vepita (is) lesser from vepaka then”
SAN: vepitaḥ = vepitaḥ + 1;
ENG: “vepita = vepita + 1;”
SAN: iti;
ENG: “thus;”

4 Conclusion and Future Work

We have tried to describe the syntax of each selected Sanskritam statements separately by giving the formal rules of each statement. The English translations of example source code may not match the exact syntax of Sanskritam specification. A complete parser of the Sanskritam specification based on a computational formal grammar starting from its start symbol will be discussed in future. Later on, we will discuss its implementation in Java environment.

Acknowledgments

We are very thankful to the Jawaharlal Nehru University for providing research infrastructure. We specially thank those friends who helped in understanding the traditional logic and Sanskrit grammar.

References

Bharati, A., Chaitanya, V., & Sangal, R. (2010). *Natural Language Processing*. New Delhi: PHI Learning Private Limited.

Bhattacharya, Kamaleswar (2006). On the Language of Navya-Nyāya: an Experiment with Precision through a Natural Language. *The Journal of Indian Philosophy, IIIIV*(1/2), 5-13.

Bhattacharya, Sibajiban (1990). Some Features of the Technical Language of Navya-Nyāya. *Philosophy East and West, XX*(2), 129-149.

Brigg, R. (1985). Knowledge Representation in Sanskrit and Artificial Intelligence. *AI Magazine*, 32-39.

Deshpande, Madhav M. (1991). Prototypes in Pāṇinian Syntax. *Journal of the American Oriental Society, III*(3), 465-480.

Huet, Gérard (2016). Sanskrit signs and Pāṇinian Scripts. *Sanskrit Computational Linguistics,* 53-76. New Delhi: D.K. Publishers.

Jha, Girish Nath (2010). *Sanskrit Computational Linguistics*. Germany: Springer Verlag.

Kadvany, J. (2016). Pāṇini's Grammar and Modern Computation. *History and Philosophy of Logic*, 325-346.

Kiparsky, P. (2002). On the Architecture of Pāṇini's Grammar. Hydrabad: Central Institute of English and Foreign Languages.

Kuhn, Tobias (2013). A Principled Approach to Grammars for Controlled Natural Languages and Predictive Editors. *Journal of Logic, Language, and Information*, *XXII*(3), 33-70.

Kuhn, Tobias (2014). A Survey and Classification of Controlled Natural Languages. *Computational Linguistics*, *XL*(1).

Parr, T. (2013). *The Definitive ANTLR 4 Reference*. Texas: Pragmatic Bookshelf.

Schildt, Herbert (2019). *Java: The Complete Reference* (11th ed.). New York: McGraw-Hill Education.

Sharma, R. N. (2002). *The AD of Panini* (2nd ed. Vol. 1). New Delhi: Munshiram Manoharlal Publishers.

Williams, Monier (1872). *A Sanskṛit-English dictionary etymologically and philologically arranged: With special reference to Greek, Latin, Gothic, German, Anglo-Saxon, and other cognate Indo-European languages*. Oxford: The Clarendon Press.