

Simulating the Paninian System of Sanskrit Grammar

Anand Mishra

► **To cite this version:**

Anand Mishra. Simulating the Paninian System of Sanskrit Grammar. First International Sanskrit Computational Linguistics Symposium, INRIA Paris-Rocquencourt, Oct 2007, Rocquencourt, France. inria-00207943

HAL Id: inria-00207943

<https://hal.inria.fr/inria-00207943>

Submitted on 18 Jan 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Simulating the Pāṇinian System of Sanskrit Grammar

Anand Mishra

Department of Computational Linguistics
Ruprecht Karls University, Heidelberg
<http://sanskrit.sai.uni-heidelberg.de>

Abstract

We propose a model for the computer representation of the Pāṇinian system of Sanskrit grammar. Based on this model, we render the grammatical data and simulate the rules of Aṣṭādhyāyī on computer. We then employ these rules for generation of morpho-syntactical components of the language. These generated components we store in a p-subsequential automata. This we use to develop a lexicon on Pāṇinian principles. We report briefly its implementation.

1 A Representation of Aṣṭādhyāyī

The general grammatical process of Aṣṭādhyāyī (Katre, 1989) can be viewed as consisting of the following three basic steps:

1. PRESCRIPTION of the fundamental components which constitute the language.
2. CHARACTERIZATION of these fundamental components by assigning them a number of attributes.
3. SPECIFICATION of grammatical operations based on the fundamental components and their attributes.

1.1 Fundamental Components

In his grammar Pāṇini furnishes a number of elements (phonemes/morphemes/lexemes) which constitute the building blocks of the language. We assign each of them a unique key in our database. Thus the phoneme /a/ has the key a_0, the *kṛt* suffix /a/ has the key a_3

and the *taddhita* suffix /a/ is represented by the key a_4. Given such a collection of unique keys, we define the set of fundamental components as follows:

Definition 1 *The collection of unique keys corresponding to the basic constituents of the language, we define as the set \mathcal{F} of fundamental components.*

Further, we decompose this set \mathcal{F} into two disjoint sets \mathcal{P} and \mathcal{M} , where \mathcal{P} is the set of keys corresponding to the phonemes and \mathcal{M} containing the keys of the rest of the constituting elements (morphemes/lexemes).

$$\mathcal{P} = \{a_0, i_0, u_0, \dots\}$$

$$\mathcal{M} = \{bhU_a, tip_0, laT_0, \dots\}$$

$$\mathcal{F} = \mathcal{P} \cup \mathcal{M} \quad (1)$$

$$\mathcal{P} \cap \mathcal{M} = \phi \quad (2)$$

1.2 Attributes

The fundamental units of the language are given an identity by assigning a number of attributes to them. The various technical terms introduced in the grammar come under this category, as also the *it* -markers and sigla or *pratyāhāras*. For example, the attributes *hrasva*, *guṇa* and *aC* characterize the element a_0 as short vowel /a/ and attributes like *pratyaya*, *prathama* and *ekavacana* tell that *tiP* (= *tip_0*) is a third-person singular suffix. Again, each attribute is assigned a unique key in our database.

Definition 2 *The collection of unique keys corresponding to the terms, which characterize a fundamental component, we define as the set \mathcal{A} of attributes.*

Corresponding to the sets \mathcal{P} and \mathcal{M} we can decompose the set \mathcal{A} into two disjoint sets \mathcal{A}_π and \mathcal{A}_μ , \mathcal{A}_π being the set of unique keys of the attributes to the elements of \mathcal{P} and \mathcal{A}_μ to elements of \mathcal{M} .

$$\begin{aligned}\mathcal{A}_\pi &= \{\text{hrasva}_0, \text{udAtta}_0, \text{it}_0, \dots\} \\ \mathcal{A}_\mu &= \{\text{dhAtu}_0, \text{pratyaya}_0, \text{zit}_9, \dots\} \\ \mathcal{A} &= \mathcal{A}_\pi \cup \mathcal{A}_\mu\end{aligned}\quad (3)$$

We note that any two of the four sets $\mathcal{P}, \mathcal{M}, \mathcal{A}_\pi, \mathcal{A}_\mu$ are mutually disjoint.

2 Basic Data Structures

Given the set of fundamental components ($\mathcal{F} = \mathcal{P} \cup \mathcal{M}$) and the set of attributes ($\mathcal{A} = \mathcal{A}_\pi \cup \mathcal{A}_\mu$), we now define our data structure for representing the Pāṇinian process.

2.1 Sound Set ψ

Definition 3 A sound set ψ is a collection of elements from sets \mathcal{P}, \mathcal{M} and \mathcal{A} having exactly one element from the set \mathcal{P} .

$$\psi = \{\pi_p, \mu_i, \alpha_j \mid \pi_p \in \mathcal{P}, \mu_i \in \mathcal{M}, \alpha_j \in \mathcal{A}, i, j \geq 0\} \quad (4)$$

This is an abstract data structure. Although it corresponds to a phoneme or one sound unit, it represents more than just a phoneme.

2.2 Language Component λ

Definition 4 A language component λ is an ordered collection of at least one or more sound sets.

$$\lambda = [\psi_0, \psi_1, \psi_2, \dots, \psi_n] \text{ such that } \|\lambda\| > 0 \quad (5)$$

CONVENTION: We use square brackets $[\]$ to represent an ordered collection and curly brackets $\{ \}$ for an unordered collection.

Language expressions at every level (phonemes, morphemes, lexemes, words, sentences) can now be represented as a language component.

Example 1: We represent the verbal root $bh\bar{u}$ as a language component λ .

$$\begin{aligned}\lambda &= [\psi_1, \psi_2] \text{ where} \\ \psi_1 &= \{bh, bh\bar{u}, dh\bar{a}tu, \dots\} \\ \psi_2 &= \{u, bh\bar{u}, dh\bar{a}tu, ud\bar{a}tta, d\bar{r}gha, aC, \dots\}\end{aligned}$$

Corresponding to the two phonemes bh and \bar{u} in $bh\bar{u}$, we have an ordered collection of two sound sets ψ_1 and ψ_2 . Consider the first one ψ_1 : Its first element bh is from the phoneme set \mathcal{P} .¹ The second element $bh\bar{u}$ tells that the phoneme of this sound set is a part of the fundamental unit $bh\bar{u}$. The third element stores the attribute $dh\bar{a}tu$ (verbal root) to this sound set. Similarly, the second sound set ψ_2 has phoneme attributes which tell it to be an $ud\bar{a}tta$ (high pitched) $d\bar{r}gha$ (long) aC (vowel).

Example 2: Similarly, the language component corresponding to the morpheme $lA\ddot{T}$ is:

$$\begin{aligned}\lambda &= [\psi] \text{ where} \\ \psi &= \{l, lA\ddot{T}, pratyaya, ait, \ddot{t}it, \dots\}\end{aligned}$$

Attribute $\ddot{t}it$ says that it has \ddot{t} as it - marker.

Example 3: The morphemes $bh\bar{u}$ followed by $lA\ddot{T}$ can now be represented together by the language component

$$\begin{aligned}\lambda &= [\psi_1, \psi_2, \psi_3] \text{ where} \\ \psi_1 &= \{bh, bh\bar{u}, dh\bar{a}tu, \dots\} \\ \psi_2 &= \{u, bh\bar{u}, dh\bar{a}tu, ud\bar{a}tta, d\bar{r}gha, aC, \dots\} \\ \psi_3 &= \{l, lA\ddot{T}, pratyaya, ait, \ddot{t}it, \dots\}\end{aligned}$$

Different linguistic units can now be identified by carrying out intersection with the appropriate subsets of \mathcal{P}, \mathcal{M} and \mathcal{A} . For example to get the verbal root or $dh\bar{a}tu$ in λ we take the intersection of an identity set $\iota = \{dh\bar{a}tu\}$ with each of ψ_i 's in λ and store the index i when the intersection-set is not empty. In this case we get the index list $[1,2]$. The list of ψ_i 's corresponding to these indices then gives the searched morpheme. Thus, the verbal root is given by the language component $[\psi_1, \psi_2]$.

2.3 Process Strip σ

Definition 5 A process strip σ is an ordered collection of pairs, where the first element of the pair is the number of a particular grammatical rule (e.g. $rule_p$) and the second element is a language component λ .

$$\sigma = [(rule_p, \lambda_p), (rule_q, \lambda_q), \dots] \quad (6)$$

The rule number corresponds to the Aṣṭādhyāyī order and binds the process

¹Actually, it is the unique key bh_0 corresponding to the phoneme bh which is stored in the sound set.

strip with a function implementing that rule in the actual program. Thus, the process strip simulates the Pāṇinian process by storing in the language component λ_p the effect of applying the rule $rule_p$.

3 Basic Operations

Having defined our data-structure, we now introduce the basic operations on them.

3.1 Attribute Addition

Let $\alpha \subset \mathcal{A} \cup \mathcal{M}$ and ψ be a sound set. Then attribute addition is defined as

$$h_{a\psi}(\psi, \alpha) = \psi \cup \alpha \quad (7)$$

This operation can be applied to a number of sound sets given by indices $[i, i+1, \dots, j]$ in a given language component λ

$$h_{a\lambda}(\lambda, \alpha, [i, \dots, j]) = [\psi_1, \dots, \psi_i \cup \alpha, \dots, \psi_j \cup \alpha, \dots, \psi_n] \quad (8)$$

Example 4: Consider the language component corresponding to the morpheme $\acute{s}aP$

$$\begin{aligned} \lambda &= [\psi] \text{ where} \\ \psi &= \{a, \acute{s}aP, pratyaya, \acute{s}it, pit, \dots\} \end{aligned}$$

RULE $ti\acute{n} \acute{s}it \acute{s}arvadh\acute{a}tukam$ (3.4.113) says that affixes in the siglum $ti\acute{N}$ and those having \acute{s} as it marker are assigned the attribute $\acute{s}arvadh\acute{a}tuka$. We implement this rule by checking if there are sound sets with attributes $pratyaya$ together with $ti\acute{N}$ or $\acute{s}it$ and adding the attribute $\acute{s}arvadh\acute{a}tuka$ if the condition is fulfilled. In this case, we get:

$$\begin{aligned} \lambda &= [\psi] \text{ where} \\ \psi &= \{a, \acute{s}aP, pratyaya, \acute{s}it, pit, \acute{s}arvadh\acute{a}tuka\} \end{aligned}$$

3.2 Augmentation

Let

$$\begin{aligned} \lambda &= [\psi_1, \dots, \psi_i, \psi_{i+1}, \dots, \psi_n] \\ \lambda_k &= [\psi_{1k}, \psi_{2k}, \psi_{3k}, \dots, \psi_{mk}] \end{aligned}$$

and i be an integer index such that $i \leq \|\lambda\|$, then augmentation of λ by λ_k at index i is defined as

$$h_g(\lambda, \lambda_k, i) = [\psi_1, \dots, \psi_i, \psi_{1k}, \psi_{2k}, \psi_{3k}, \dots, \psi_{mk}, \psi_{i+1}, \dots, \psi_n] \quad (9)$$

Example 5: Consider the language component λ corresponding to the verbal root $bh\bar{u}$.

$$\begin{aligned} \lambda &= [\psi_1, \psi_2] \text{ where} \\ \psi_1 &= \{bh, bh\bar{u}, dh\acute{a}tu, \dots\} \\ \psi_2 &= \{u, bh\bar{u}, dh\acute{a}tu, ud\acute{a}tta, d\bar{ir}gha, aC, \dots\} \end{aligned}$$

RULE $vartam\acute{a}ne la\ddot{T}$ (3.2.123) says that the morpheme $la\ddot{T}$ is added after a $dh\acute{a}tu$ if the present action is to be expressed. To implement this rule, we first look for the indices of sound sets which have the attribute $dh\acute{a}tu$ and then append the sound set corresponding to $la\ddot{T}$ after the last index. We get,

$$\begin{aligned} \lambda &= [\psi_1, \psi_2, \psi_3] \text{ where} \\ \psi_1 &= \{bh, bh\bar{u}, dh\acute{a}tu, \dots\} \\ \psi_2 &= \{u, bh\bar{u}, dh\acute{a}tu, ud\acute{a}tta, d\bar{ir}gha, aC, \dots\} \\ \psi_3 &= \{l, la\ddot{T}, pratyaya, ait, \acute{t}it, \dots\} \end{aligned}$$

3.3 Substitution

We define substitution in terms of the above two operations.

Let $[i, i+1, i+2, \dots, j]$ be the indices of sound sets to be replaced in the language component $\lambda = [\psi_1, \dots, \psi_i, \psi_{i+1}, \dots, \psi_n]$.

Let $\lambda_k = [\psi_{1k}, \psi_{2k}, \psi_{3k}, \dots, \psi_{mk}]$ be the replacement, then the substitution is defined as

$$h_s(\lambda, \lambda_k, [i, \dots, j]) = h_g(h_{a\lambda}(\lambda, \{\delta\}, [i, \dots, j]), \lambda_k, j) \quad (10)$$

where $\delta \in \mathcal{A}$ is the *attribute* which says that this sound set is no more active and has been replaced by some other sound set.

Example 6: Consider the language component corresponding to the verbal root $\eta\bar{i}\tilde{N}$

$$\begin{aligned} \lambda &= [\psi_1, \psi_2] \text{ where} \\ \psi_1 &= \{\eta, \eta\bar{i}\tilde{N}, dh\acute{a}tu, \tilde{n}it\} \\ \psi_2 &= \{i, \eta\bar{i}\tilde{N}, dh\acute{a}tu, \tilde{n}it, d\bar{ir}gha, aC\} \end{aligned}$$

RULE $\eta\acute{h} \eta\acute{h}$ (6.1.065) says that the initial retroflex η of a $dh\acute{a}tu$ is replaced by dental n . To implement this rule we first search the sound sets corresponding to $dh\acute{a}tu$, check whether the first one has a retroflex η and if the conditions are fulfilled, add the attribute δ in that sound set and append the sound set corresponding to n after it. Further we

transfer all attributes (except the phoneme attributes) from the η - sound set to n - sound set for *sthānivadbhāva*. We get,

$$\begin{aligned}\lambda &= [\psi_1, \psi_2, \psi_3] \text{ where} \\ \psi_1 &= \{n, \eta\tilde{N}, dhātu, \tilde{n}it, \delta\} \\ \psi_2 &= \{n, \eta\tilde{N}, dhātu, \tilde{n}it\} \\ \psi_3 &= \{i, \eta\tilde{N}, dhātu, \tilde{n}it, \tilde{d}irgha, aC\}\end{aligned}$$

4 Grammatical Process

4.1 Representing a Rule of Grammar

We represent a rule of grammar through a function f_q , which takes a process strip σ_p and adds a new pair $(rule_q, \lambda_q)$ to it where $rule_q$ is the number of the present rule and λ_q is the new modified language component after application of one or more of the three operations defined above on the input language component λ_p .

$$\begin{aligned}f_q(\sigma_p) &= \sigma_q \text{ where} \\ \sigma_p &= [\dots, (rule_p, \lambda_p)] \\ \sigma_q &= [\dots, (rule_p, \lambda_p), (rule_q, \lambda_q)] \\ \lambda_q &= h_a, h_g, h_s(\lambda_p, \dots)\end{aligned}$$

4.2 Structure of a rule

The general structure of a rule is as follows:

Function f_q with input strip:

$$\sigma_p = [\dots, (rule_p, \lambda_p)]$$

check applicability conditions
if conditions not fulfilled then
return unextended σ_p
else

create new modified λ_q
return extended σ_q

Thus, given a particular state (represented by σ_p) in the process of generation, the system provides for checking the applicability of a rule f_q , and if the conditions are fulfilled, the rule is applied and the changed language component together with the rule number is stored in the modified state (represented by σ_q).

As the rule numbers are also stored, we can implement the rules of *tripādī* and make their effects invisible for subsequent applications. The order in which rules are applied is provided manually through templates.

5 Example

We take a verbal root *bhū* and generate the final word *bhavati* meaning “he/she/it becomes”. We initialize the process strip σ_0 by loading the language component corresponding to the verbal root and adding *a00000* as the rule number.

$$\begin{aligned}\sigma_0 &= [(a00000, \lambda_0)] \text{ where} \\ \lambda_0 &= [\psi_{0a}, \psi_{0b}] \text{ with} \\ \psi_{0a} &= \{bh, bhū, dhātu\} \\ \psi_{0b} &= \{u, bhū, dhātu, \tilde{d}irgha, udātta\}\end{aligned}$$

RULE *vartamāne laṭ* (3.2.123) says that the morpheme *laṭ* is added after a *dhātu* if the present action is to be expressed. The application now involves following steps: Look in the last language component λ of the process-strip σ . If there are sound sets ψ with the identity set $\iota = \{dhātu\}$ in it, get their indices in index list. This returns the index list [1,2]. If index list is non empty then augment the language component λ_0 by attaching the language component corresponding to the morpheme *laṭ*. This is attached in this case at index 2 as the new morpheme comes after *dhātu*. Extend the process strip σ_0 accordingly.

$$f_{a32123}(\sigma_0) = \sigma_1$$

$$\begin{aligned}\sigma_1 &= [(a00000, \lambda_0), (a32123, \lambda_1)] \text{ where} \\ \lambda_1 &= [\psi_{0a}, \psi_{0b}, \psi_{1a}] \text{ with} \\ \psi_{0a} &= \{bh, bhū, dhātu\} \\ \psi_{0b} &= \{u, bhū, dhātu, \tilde{d}irgha, udātta\} \\ \psi_{1a} &= \{l, laṭ, pratyaya, ait, \tilde{t}it\}\end{aligned}$$

RULE *tip tas jhi sip thas tha mip vas mas ta ātām jha thās āthām dhvam iṭ vahi mahiñ* (3.4.078) provides for substitution of *laṭ*. We take the first suffix *tiP* for replacement. The sound sets to be replaced are determined by taking intersection with the set $\{laṭ, liṭ, loṭ, \dots\}$ which has the morphemes having cover term *l*. In this case it is at the index 3. We replace this sound set with *tiP* i.e. add the attribute δ to the sound set at index 3 and augment the language component at this index.

$$f_{a34078}(\sigma_1) = \sigma_2$$

$$\begin{aligned}
\sigma_2 &= [\dots, (a32123, \lambda_1), (a34078, \lambda_2)] \\
\lambda_2 &= [\psi_{0a}, \psi_{0b}, \psi_{1a}, \psi_{2a}, \psi_{2b}] \\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu\} \\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, d\bar{i}rgha, ud\bar{a}tta\} \\
\psi_{1a} &= \{l, lA\ddot{T}, pratyaya, ait, \ddot{t}it, \delta, \} \\
\psi_{2a} &= \{t, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\} \\
\psi_{2b} &= \{i, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

RULE kartari śap (3.1.068) says that the morpheme śaP is added after *dhātu* but before *sārvadhātuka* suffix and denotes agent. Check if sound set with *sārvadhātuka* follows one with *dhātu*. If yes then augment the language component for śaP after *dhātu*.

$$f_{a31068}(\sigma_2) = \sigma_3$$

$$\begin{aligned}
\sigma_3 &= [\dots, (a34078, \lambda_2), (a31068, \lambda_3)] \\
\lambda_3 &= [\psi_{0a}, \psi_{0b}, \psi_{3a}, \psi_{1a}, \psi_{2a}, \psi_{2b}] \\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu\} \\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, d\bar{i}rgha, ud\bar{a}tta\} \\
\psi_{3a} &= \{a, śaP, pratyaya, hrasva, śit, pit\} \\
\psi_{1a} &= \{l, lA\ddot{T}, pratyaya, ait, \ddot{t}it, \delta, \} \\
\psi_{2a} &= \{t, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\} \\
\psi_{2b} &= \{i, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

RULE yasmāt pratyaya vidhis tad ādi pratyaye aṅgam (1.4.013) makes the part before the suffix śaP an *aṅga* with respect to it.

$$f_{a14013}(\sigma_3) = \sigma_4$$

$$\begin{aligned}
\sigma_4 &= [\dots, (a31068, \lambda_3), (a14013, \lambda_4)] \\
\lambda_4 &= [\psi_{0a}, \psi_{0b}, \psi_{3a}, \psi_{1a}, \psi_{2a}, \psi_{2b}] \\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu, aṅga\} \\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, aṅga, d\bar{i}rgha, ud\bar{a}tta\} \\
\psi_{3a} &= \{a, śaP, pratyaya, hrasva, śit, pit\} \\
\psi_{1a} &= \{l, lA\ddot{T}, pratyaya, ait, \ddot{t}it, \delta, \} \\
\psi_{2a} &= \{t, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\} \\
\psi_{2b} &= \{i, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

RULE sārva dhātuka ārdhadhātukayoh (7.3.084) says that before *sārvadhātuka* or *ārdhadhātuka* replace the *i*K vowels by *guṇa* vowels. As śaP is *sārvadhātuka*, we get

$$f_{a73084}(\sigma_4) = \sigma_5$$

$$\begin{aligned}
\sigma_5 &= [\dots, (a14013, \lambda_4), (a73084, \lambda_5)] \\
\lambda_5 &= [\psi_{0a}, \psi_{0b}, \psi_{5a}, \psi_{3a}, \psi_{1a}, \psi_{2a}, \psi_{2b}] \\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu, aṅga\} \\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, aṅga, d\bar{i}rgha, ud\bar{a}tta, \delta\} \\
\psi_{5a} &= \{o, bh\bar{u}, dh\bar{a}tu, aṅga\} \\
\psi_{3a} &= \{a, śaP, pratyaya, hrasva, śit, pit\} \\
\psi_{1a} &= \{l, lA\ddot{T}, pratyaya, ait, \ddot{t}it, \delta, \} \\
\psi_{2a} &= \{t, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\} \\
\psi_{2b} &= \{i, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

RULE ec aḥ ay av āy āv aḥ (6.1.078) says that before *a*C (vowel) *e*, *o*, *ai*, *au* are respectively replaced by *ay*, *av*, *āy*, *āv*.

$$f_{a61078}(\sigma_5) = \sigma_6$$

$$\begin{aligned}
\sigma_6 &= [\dots, (a73084, \lambda_5), (a61078, \lambda_6)] \\
\lambda_6 &= [\psi_{0a}, \psi_{0b}, \psi_{5a}, \psi_{6a}, \psi_{6b}, \psi_{3a}, \psi_{1a}, \psi_{2a}, \psi_{2b}] \\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tuaṅga\} \\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, aṅga, d\bar{i}rgha, ud\bar{a}tta, \delta\} \\
\psi_{5a} &= \{o, bh\bar{u}, dh\bar{a}tu, aṅga, \delta\} \\
\psi_{6a} &= \{a, av, bh\bar{u}, dh\bar{a}tu, aṅga, hrasva\} \\
\psi_{6b} &= \{v, av, bh\bar{u}, dh\bar{a}tu, aṅga\} \\
\psi_{3a} &= \{a, śaP, pratyaya, hrasva, śit, pit\} \\
\psi_{1a} &= \{l, lA\ddot{T}, pratyaya, ait, \ddot{t}it, \delta, \} \\
\psi_{2a} &= \{t, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\} \\
\psi_{2b} &= \{i, tiP, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

Finally we collect all ψ_i s not having a δ , i.e. which are not already replaced. This gives us the desired form *bhavati*.

6 PaSSim (Pāṇinian Sanskrit Simulator)

In the following we give a brief description of PaSSim (Pāṇinian Sanskrit Simulator) we are developing at the University of Heidelberg.² The program aims towards developing a lexicon on Pāṇinian principles. The user enters an inflected word or *pada* and the system furnishes a detailed, step by step process of its generation. It is written in PythonTM and consists of the following modules (See Figure 1):

6.1 Database

This module is for inputting, updating, enhancing and organizing the primary database

²<http://sanskrit.sai.uni-heidelberg.de>

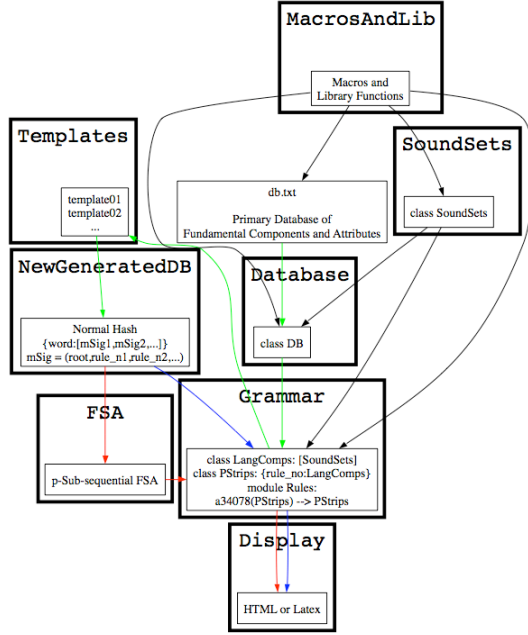


Figure 1: PaSSim (Pāṇinian Sanskrit Simulator)

of fundamental components and attributes. The organization of database serves the purpose of incorporating static information of Pāṇinian formulations. For example, $u\check{N}$ is stored with *static* attributes *dhātu*, *bhvādi*, *aniṭ* and that its second phoneme is *it* - marker etc. Thus, the effect of many definition rules of Aṣṭādhyāyī are stored in the database. The database is in ASCII and each fundamental component or attribute has a unique key corresponding to which is a hash.

6.2 Grammar

This is the main module. It contains abstract classes corresponding to `SoundSets`, `LanguageComponents` and `ProcessStrips`. Further it has a number of functions like `a61065()`, which simulate the individual rules of Aṣṭādhyāyī.

6.3 Templates

This module is to organize the *prakriyā*. A `template` prescribes the rules in order of applicability for a group of primary verbs or nominal stems. Templates are specified manually, taking into account the *prakriyā* texts e.g. Siddhānta-Kaumudī (Vasu, 1905).³ It uses

³We would like to acknowledge two texts in Hindi — Vyākaraṇacandrodaya (Śāstrī, 1971) and Aṣṭādhyāyī

Grammar to generate the morpho-syntactic word forms or *padas*.

6.4 FSA

This module is for the sake of efficient representation of generated words together with the initializing fundamental component(s) and list of rule numbers. These are stored as a p-subsequential transducer (Mohri, 1996).⁴ The output string associated with a word, thus provides the initializing fundamental components and a list of rules. Grammar applies these rules one after another and outputs the final as well as intermediate results.

6.5 Display

This module provides HTML⁵ / L^AT_EX output. It outputs the content according to the given style sheet for conventions regarding script, color-scheme etc. The phonological, morphological, syntactical and semantical information gathered during the process of generation is rendered in modern terms through a mapping of Pāṇinian attributes corresponding to it.

References

- Böhtlingk, Otto von. 1887. *Pāṇini's Grammatik*. Olms, Hildesheim. Primary source text for our database.
- Dikṣita, Puṣpā. 2006-07. *Aṣṭādhyāyī sahajabodha*. Vols. 1-4. Pratibha Prakashan, Delhi, India.
- Katre, Sumitra M. 1989. *Aṣṭādhyāyī of Pāṇini*. Motilal Banarsidass, Delhi, India.
- Mohri, Mehryar. 1996. On some Applications of Finite-State Automata Theory to Natural Language Processing. *Journal of Natural Language Engineering*, 2:1-20.
- Śāstrī, Cārudeva. 1971. *Vyākaraṇacandrodaya*. Vols. 1-5. Motilal Banarsidass, Delhi, India.
- Vasu, Srisa Chandra and Vasu, Vaman Dasa. 1905. *The Siddhānta-Kaumudī of Bhaṭṭojī Dikṣita*. Vols. 1-3. Panini Office, Bhuvaneshvara Asrama, Allahabad, India. Primary source text for *prakriyā*.

sahajabodha (Dikṣita, 2006) — which have been very beneficial to us.

⁴Sequential transducers can be extended to allow a single additional output string (subsequential transducers) or a finite number p of output strings (p - subsequential transducers) at final states. These allow one to deal with the ambiguities in natural language processing (Mohri, 1996).

⁵See: <http://sanskrit.sai.uni-heidelberg.de>