Analysis of Sanskrit Compounds: Constituency or Dependency?

Amba Kulkarni[1]

Department of Sanskrit Studies
University of Hyderabad

LINGUINDIC
Oxford University
June 12-14, 2025

[1] with major contributions by Pavankumar Satuluri and Sriram Krishnan

Outline

Background

Intriguing Questions

- How do humans express their thoughts through linguistic expressions?
- How does a listener interpret these thoughts?

Intriguing Questions

- How do humans express their thoughts through linguistic expressions
  $\Rightarrow$ Grammar
- How does a listener interpret these thoughts
  $\Rightarrow$ Theories of verbal cognition

Śābdabodha: Essential factors

Kumārila Bhaṭṭa in his *Tantravārttika* mentions 3 factors necessary for the understanding of the correlation between words.[2] – (Vol I p. 455)
These factors are

- Ākāṅkṣā (expectancy)
- Yogyatā (meaning congruity)
- Sannidhi (proximity)

[2] *Ākāṅkṣā sannidhānaṁ ca yogyatā ceti ca trayam sambandhakāraṇatvena kḷptaṁ nānantaraśrutiḥ.*

Dependency Analysis

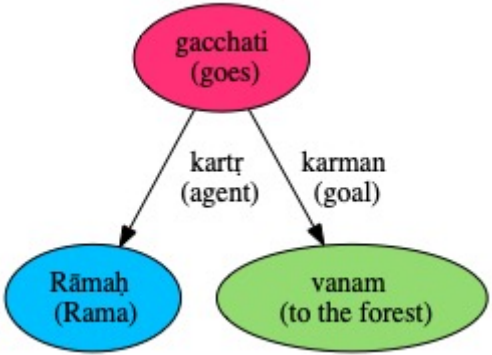


Figure: Directed Acyclic Graph

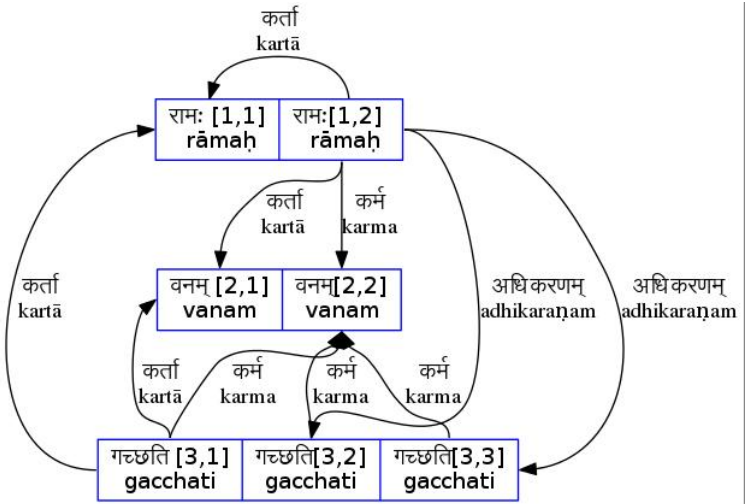
Dependency Parser

Parser - Algorithm

1. Define nodes
2. Establish directed edges
3. Define the constraints
   1. Local constraints:
      1. A morpheme corresponding to a suffix marks only one relation.
      2. Each *kāraka* relation is marked by a single morpheme.
      3. A morpheme does not mark a relation to self.
      4. There can be only one valid analysis of every word per solution.
   2. Global Constraints:
      The *sannidhi* constraint ensures that there are no crossing of edges.
4. Extract a Tree from the Graph, satisfying the constraints.
5. Rank the solutions.

Parser - An illustration

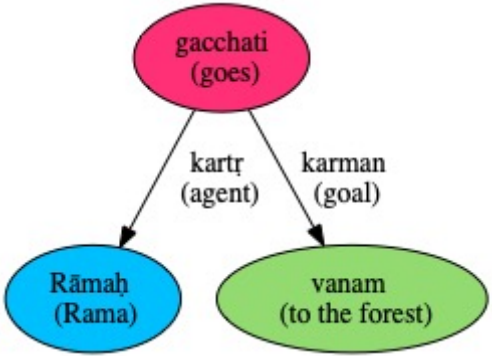
A graph showing the application of ākāṅkṣā:



Parser - An illustration – contd

After the application of constraints:

- A morpheme corresponding to a suffix marks only one relation.
- Each *kāraka* relation is marked by a single morpheme.
- A morpheme does not mark a relation to self.
- There can be only one valid analysis of every word per solution.

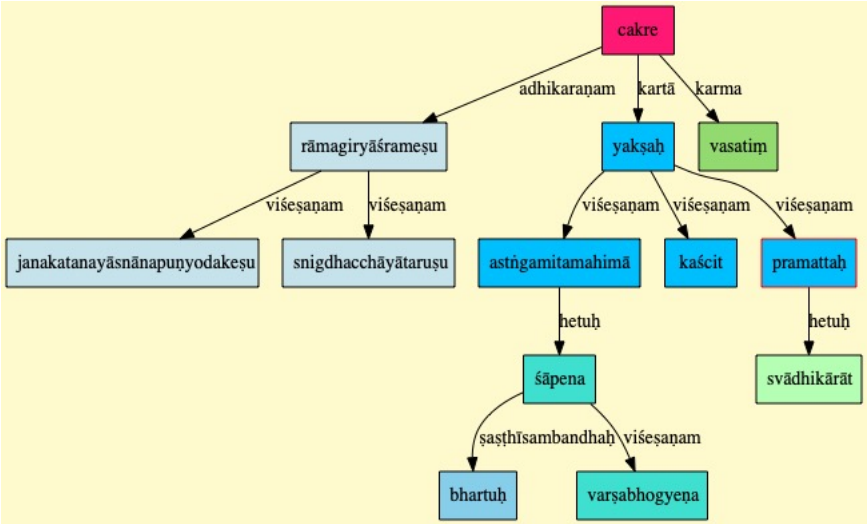


Compounds

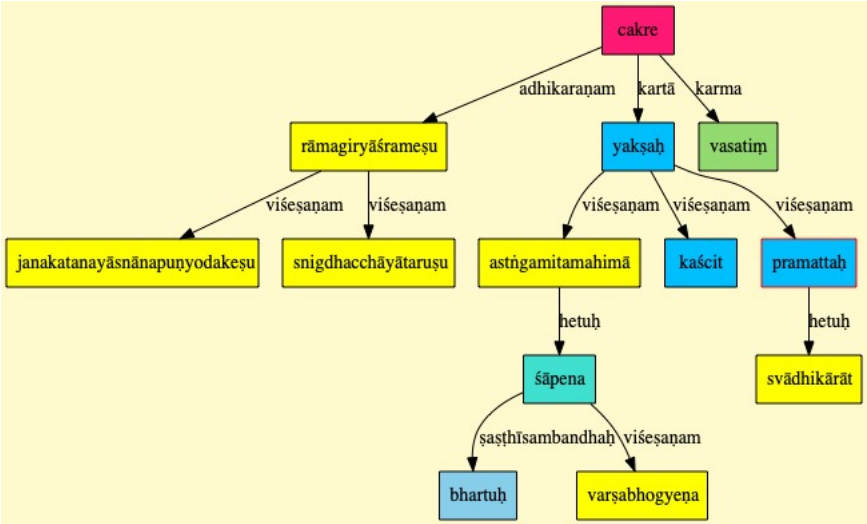
Compounds

- Very frequent
- Every fifth word is a compound.
- Without understanding the meaning of a compound, one cannot understand the meaning of a sentence / verse.

Compound Analysis - Necessity



Compound Analysis - Necessity



Compound Analysis

Compound Analysis - Steps

1. Segmentation,
2. Constituency Analysis,
3. Compound Type Identification, and
4. Generating a Paraphrase.

Compound Analysis - Steps

1. Segmentation: *tapas-sva-adhyāya-niratam*
2. Constituency Analysis: *<<tapas-<sva-adhyāya>>-niratam>*
3. Compound Type Identification:
   <<tapas-<sva-adhyāya> $T6$ *> Di-niratam>* $T7$
   where T6 and T7 stand for *Ṣaṣṭhī* and *saptamī Tatpuruṣaḥ* respectively, and , Di for *Itaretara Dvandvaḥ*.
4. Paraphrase:
   svasya adhyāyaḥ = svādhyāyaḥ (self study)
   tapas ca svādhyāyaḥ ca = tapassvādhāyau (penance and self-study)
   tapassvādhyāyayoḥ nirataḥ = tapassvādhyāyaniratah (constantly engaged in the penance and self-study)
   Since the original compound is in accusative case,
   tam = tapassvādhyāyaniratam (to who is constantly engaged in the penance and self-study)

Earlier Work

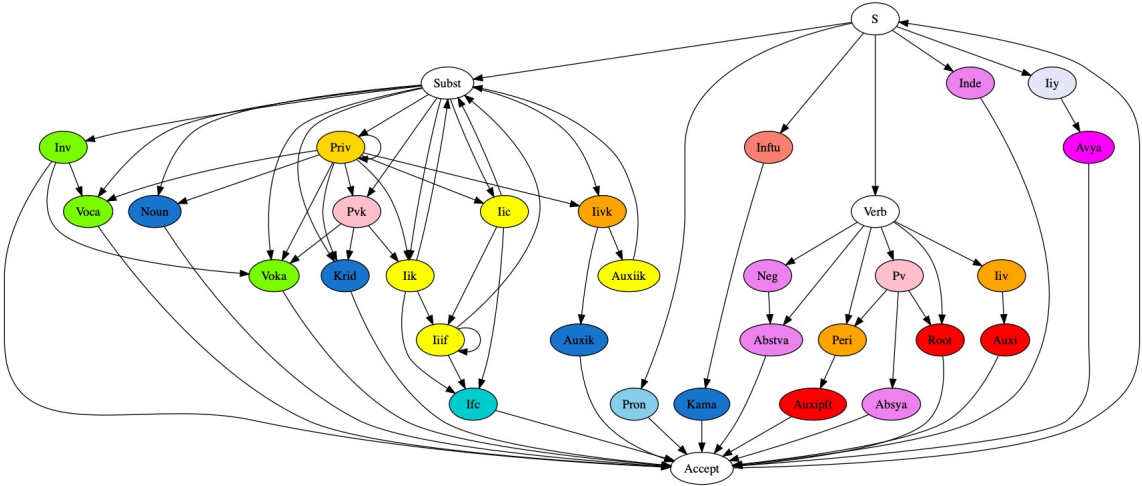
Anilkumar - PhD Thesis – provided a generalised framework

Earlier Work ... Contd

Heritage Platform

- Segmentation - identify compound boundaries
- Type identification - with POS information

Fine grain POS for segmentation



Paraphrase Generation

- *kaṣṭaśritaḥ* = kaṣṭam śritaḥ
- corabhayam = corāt bhayam
- rājapuruṣaḥ = rājñaḥ puruṣaḥ

Fine grain Type Classification

compound type:	Avyayībhāvaḥ
avyaya-pūrvapadaḥ	A1
avyaya-uttarapadaḥ	A2
tiṣṭhadguprabhṛti	A3
saṃkhyāpūrvapada-nadyuttarapadaḥ	A4
nadyuttarapada-anyapadārtha-saṃjñāyām	A5
saṃkhyāpūrvapada-vaṃśyottarapadaḥ	A6
pāre-madhye-pūrvapada ṣaṣṭhyuttarapadaḥ	A7

Fine grain Type Classification ... Contd

compound type:	Karmadhārayaḥ
viśeṣaṇa-pūrvapada-karmadhārayaḥ	K1
viśeṣaṇa-uttarapada-karmadhārayaḥ	K2
viśeṣaṇa- ubhayapada-karmadhārayaḥ	K3
upamāna-pūrvapada-karmadhārayaḥ	K4
upamāna-uttarapada-karmadhārayaḥ	K5
avadhāraṇāpūrvapada-karmadhārayaḥ	K6
sambhāvanāpūrvapada-karmadhārayaḥ	K7
madhyamapadalopī-karmadhārayaḥ	Km

Fine grain Type Classification ... Contd

compound type:	Tatpuruṣaḥ
prathamātatpuruṣaḥ	T1
dvitīyātatpuruṣaḥ	T2
tṛtīyātatpuruṣaḥ	T3
caturthītatpuruṣaḥ	T4
pañcamītatpuruṣaḥ	T5
ṣaṣṭhītatpuruṣaḥ	T6
saptamītatpuruṣaḥ	T7
nañtatpuruṣaḥ	Tn
samāhāra-dviguḥ	Tds
taddhitārthadviguḥ	Tdt
uttarapadadviguḥ	Tdu

Fine grain Type Classification ... Contd

compound type:	Tatpuruṣaḥ
gatisamāsaḥ	Tg
kusamāsaḥ	Tk
prādisamāsaḥ	Tp
mayūravyaṃsakādiḥ	Tm
bahupada-tatpuruṣaḥ	Tb
upapada-tatpuruṣaḥ	U

Fine grain Type Classification ... Contd

samānādhikaraṇabahuvrīhiḥ

compound type:	Bahuvrīhiḥ
dvitīyārtha-bahuvrīhiḥ	Bs2
tṛtīyārtha-bahuvrīhiḥ	Bs3
caturthyartha-bahuvrīhiḥ	Bs4
pañcamyartha-bahuvrīhiḥ	Bs5
ṣaṣṭhyartha-bahuvrīhiḥ	Bs6
saptamyartha-bahuvrīhiḥ	Bs7
digvācaka-bahuvrīhiḥ	Bsd
praharaṇaviṣayaka-bahuvrīhiḥ	Bsp
grahaṇaviṣayaka-bahuvrīhiḥ	Bsg
astyartha-madhyamapadalopī(nañ)-bahuvrīhiḥ	Bsmn
prādi-bahuvrīhiḥ	Bvp
saṃkhyobhayapada-bahuvrīhiḥ	Bss
upamānapūrvapada-bahuvrīhiḥ	Bsu

Fine grain Type Classification ... Contd

vyadhikaraṇabahuvrīhiḥ

compound type:	Bahuvrīhiḥ
vyadhikaraṇa-bahuvrīhiḥ	Bv
saṃkhyottarapada-vyadhikaraṇa-bahuvrīhiḥ	Bvs
sahapūrvapada-vyadhikaraṇa-bahuvrīhiḥ	BvS
upamānapūrvapada-vyadhikaraṇa-bahuvrīhiḥ	BvU
bahupada-bahuvrīhiḥ	Bb

Fine grain Type Classification ... Contd

compound type:	Dvandvaḥ
itaretarayoga-dvandvaḥ	Di
samāhāra-dvandvaḥ	Ds
ekaśeṣaḥ	E
kevalasamāsaḥ	S
dviruktiḥ	d

Earlier Work ... Contd

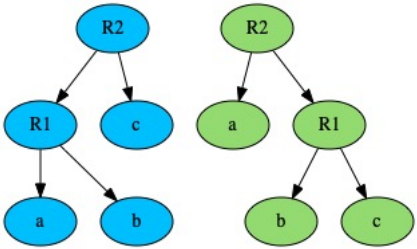
- A Novel Multi-Task Learning Approach for Context-Sensitive Compound Type Identification in Sanskrit, Oct 2022
  Jivnesh Sandhan, Ashish Gupta, Hrishikesh Terdalkar, Tushar Sandhan, Suvendu Samanta, Laxmidhar Behera, and Pawan Goyal
- DepNeCTI: Dependency-based Nested Compound Type Identification for Sanskrit, Dec 2023
  Jivnesh Sandhan, Yaswanth Narsupalli, Sreevatsa Muppirala, Sriram Krishnan, Pavankumar Satuluri, Amba Kulkarni, and Pawan Goyal.

Challenges: Constituency Analysis

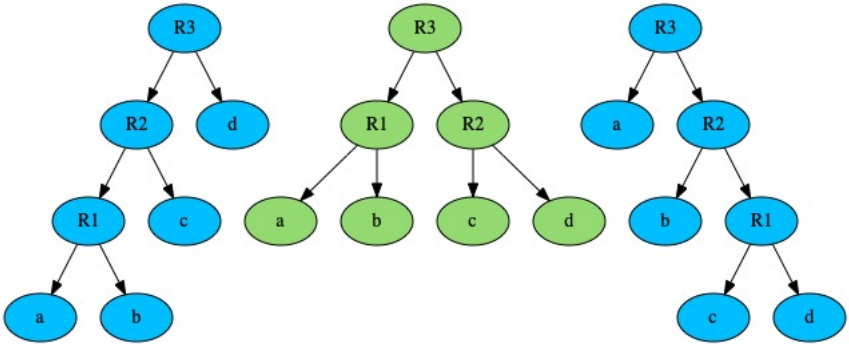
Challenges: Constituency Analysis

Determining the sāmarthya

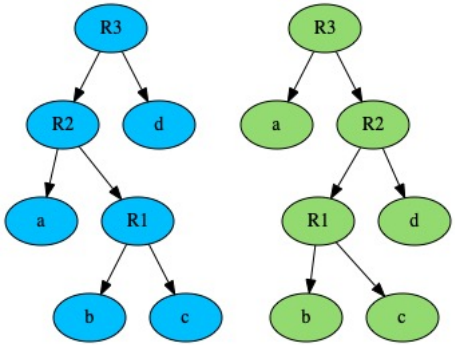
Different ways of grouping the components a-b-c?



Challenges ... Contd



Challenges ... Contd



Challenges ... Contd

Total number of possible Trees with $n+1$ components:

Catalan Number $C_n = \frac{1}{n-1} C_n^{2n}$

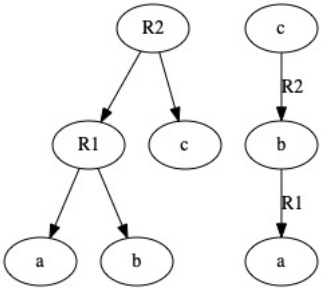
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, ...

Exceptions:

- Bahupada Bahuvrīhi and Dvandva
- The distance between two components – not more than 4 (Empirical data)

Constituency Versus Dependency

Features	Constituency	Dependency
Nodes	$2n-1$	n
Edges	$2n-2$	$n-1$
Edge-labels	No	Yes
Hierarchy	Perceived	Not perceived
Head	Not Marked	Marked



Dependency Representation of a constituency

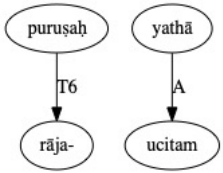
Choose Dependency – Prolixity (lāghava)

Can we faithfully represent Constituency analysis as a DAG?

Dependency Representation of a constituency ... Contd

What about the direction of the edges?
Should we mark the 'pradhāna'?

Tatpuruṣa – uttara-padārtha-pradhāna
Avyayībhāva – pūrva-padārtha-pradhāna



Dependency Representation of a constituency ... Contd

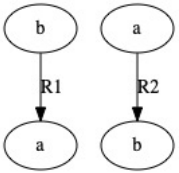


Figure: direction of arrow matters

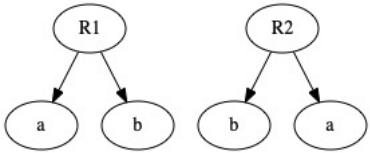


Figure: order of the components matters

Dependent - Dependee $\equiv$ component-order

Dependency Representation of a constituency ... Contd

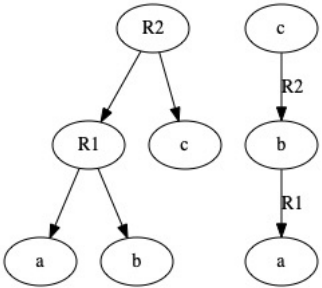


Figure: Equivalent Trees - type 1

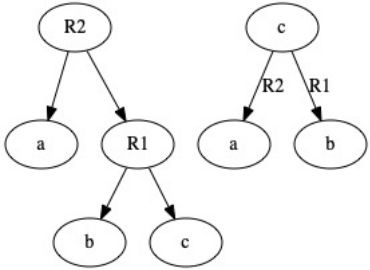


Figure: Equivalent Trees - type 2

Dependency Representation of a constituency ... Contd

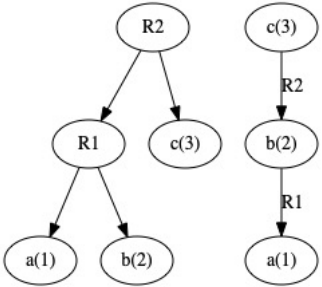


Figure: Type-1 with word-index

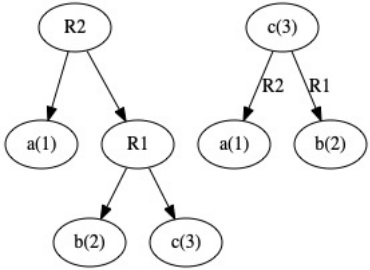


Figure: Type-2 with word-index

Constituency to Dependency

Constituency to Dependency Conversion

N: Visit the root node.

push the root node label to the Root-Label-Stack.

go to L

go to R

L: Visit the Left node

If it is a leaf node:

increment the index by 1,

push the index, node label and the root label to the dep-repre-stack

If it is an intermediate node:

go to N

... Contd

Constituency to Dependency ... Contd

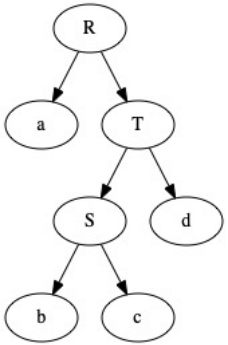
R: Visit the Right Node.
If it is a leaf node:
increment the current index,
while the root label is same as the top of Root-Label-Stack
pop up the entry from the dep-repre-stack,
pop up the entry from the Root-Label-Stack,
new root label is set to the top of the Root-Label-Stack,
append the current index to this entry
add this entry to the output table
if Root-Label-Stack is empty:
push the current index, and node label to the output table.
else:
get the root label from the Root-Label-Stack
push the current index, node label and the root label
to the dep-repre-stack.

Constituency to Dependency

If it is an intermediate node:
go to N
If there are no more nodes, then pop up the dep-repre-stack.

An illustration

Root:	R		
	Left:	a(1)	
	Right:	T	
		Left:	S
			Left: b(2)
			Right: c(3)
		Right:	d(4)



1	a	R	4
2	b	S	3
3	c	T	4
4	d		

Figure: Constituency and Dependency representations

Example

step	action	index	R-L-S	root label	D-R-S	output
1	N		R	R		
2	L	1	R	R	1 a R	
3	R					
4	N		T R	T		
5	L					
6	N		S T R	S		
7	L	2			2 b S 1 a R	

Example ... Contd

step	action	index	R-L-S	root label	D-R-S	output
8a	R	3	T		1 a R	2 b S 3
			R			
8b		3	T	T	3 c T	
		3	R		1 a R	
9a	R	4	R	R	1 a R	2 b S 3
		4	R	R	1 a R	3 c T 4
9b		4		R		2 b S 3
		4				3 c T 4
		4				1 a R 4
9c						2 b S 3
						3 c T 4
						1 a R 4
						4 d

Dependency to Constituency

For each row in the Dependency representation table do
    Store the contents of the four columns as
        Curr-From-index, Curr-From-label, Curr-Rel, and Curr-To-index
    If the Curr-To-index is undefined, skip
    If the Curr-To-index - Curr-From-index = 1:
        If the stack is non-empty:
            a) pop-up all the entries (From-index, Rel, To-index),
                where To-index = Curr-From-index

Dependency to Constituency ... Contd

b) For each such entry, construct a binary tree with a root node Rel
if Map(From-index) defined:
Left = Map(From-index)
else: Left = From-index
if Map(To-index) defined:
Right = Map(To-index)
else: Right = To-index
Map(To-index) = Rel
c) construct-a-binary-tree with current entry parameters
If the stack is empty:
construct-a-binary-tree with current entry parameters
If the Curr-To-index - Curr-From-index > 1 then
Push the current entry to the stack.
Pop-up all the entries from the stack and construct-a-binary-tree with current entry parameters

An illustration

Step	Current Entry	Stack before	Action	Stack after
1	1 a R 4	-	push to the stack	1 a R 4
2	2 b S 3	1 a R 4	Root: S Left: b(2) Right: c(3)	1 a R 4
3	3 b T 4	1 a R 4	Root: T Left: S Right: d(4)	1 a R 4
4	4 d			
		Stack pop up		
			Root: R Left: a(1) Right: T	

Table: Dependency to Constituency conversion

Compound Analyser

Compound Analyser - Algorithm

1. Define nodes
2. Establish directed edges
3. Define the constraints
4. Extract a Tree from the Graph, satisfying the constraints.
5. Rank the solutions.

Compound Analyser - Baseline Model

If there are no clues to draw the edges,

$\Rightarrow$ **n!*k** edges,

where n - nodes, and

k - number of possible relations!

Compound Analyser - Clues

- What are the clues for establishing the edges?
  - Ākāṅkṣā
  - Yogyatā
- What should be the names of the relations between the nodes?
- What is the nature of the Constraints?
  Sannidhi constraint ensures that there are no crossing of edges.

Ākāṁkṣā

2-1-49 *pūrvakālaikasarvajaratpurāṇanavakevalāḥ samānādhikaraṇena*

iic: eka, sarva, jarat, purāṇa, nava, kevala

pūrvakāla - pūrvakāla
X-*kta*-Y-*kta*

Yogyatā

Credits: Dr Anilkumar and Dr Pavankumar Satuluri - PhD dissertations

nadīvācī	deśavācī	grāmavācī
devatāvācī	ṛṣi	nakṣatravācī
saṅkhyāvācī	saṅkhyeyavācī	pūraṇa
catuṣpādavācī	varṇavācī	kṣudrajantu
vṛkṣavācī	tṛṇavācī	janapada
grantha	kāla	svāṅga
guṇavācī	dūrārtha	parimāṇa
dik	prāṇi	caraṇa
āyāmavācī	ṛkvācakāḥ	pūrayitavyavācī
vidyāvācī	dhyānavācī	yonivācī
vyañjanavācī	bhakṣyavācī	vayas

Yogyatā ... Contd

ṛṇa	niravasitāḥ	jāti
saṁjñāvācī	kāravācī	praśaṁsāvacana
adhvaryukratuḥ	sādṛśye	kṣepe
atyantasaṁyoge	adhikārārthavacana	upamānavācakaḥ
pūjā	pūjyamāna	adhikaraṇavācī
ekadeśi	karmavyatihāraḥ	adarśanam
vandanam	lubdhayogaḥ	viśeṣaṇavācī
anumeyaḥ	viprakṛṣṭākhyā	avasthā
mitra	amitra	ākroṣe

Yogyatā ... Contd

- some components can only serve as iics, for example a-/an-/su-/ku-
- some components can only serve as ifcs, for example, ambaraḥ

Constraints

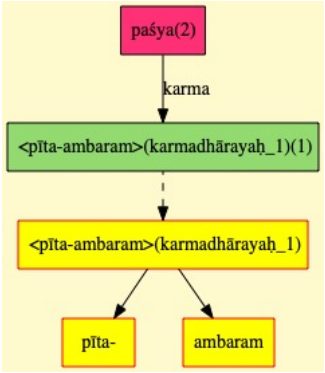
1. A component can be a pūrvapada for only one compound.
2. All the arrows are directed from right to left.
3. There can be only one valid analysis of every component per solution.
4. *sannidhi* constraint – there are no crossing of edges.

Illustration - 1

pīta-ambaram

1	pīta	ṣaṣṭhī-tatpuruṣaḥ,2 / bahuvrīhiḥ,2
2	ambaram	

... Contd


paśya(2)
karma
<pīta-ambaram>(karmadhārayaḥ_1)(1)
<pīta-ambaram>(karmadhārayaḥ_1)
pīta-
ambaram

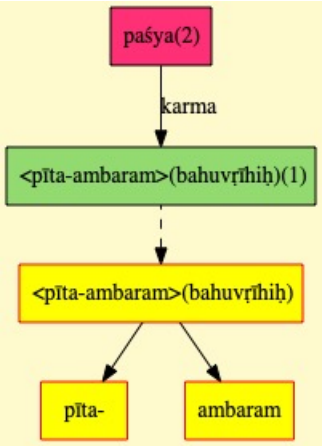

paśya(2)
karma
<pīta-ambaram>(bahuvṛīhiḥ)(1)
<pīta-ambaram>(bahuvṛīhiḥ)
pīta-
ambaram

Illustration - 2

pīta-ambaram viṣṇum paśya

1	pīta	bahuvrīhiḥ,2
2	ambaram	viśeṣaṇam
3	viṣṇum	karma,4
4	paśya	

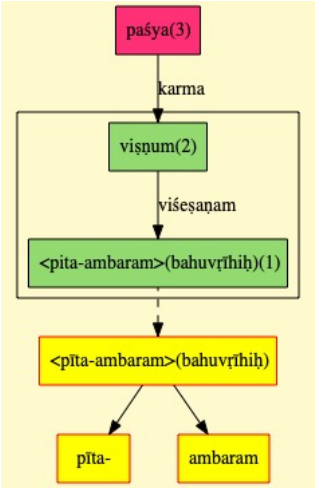
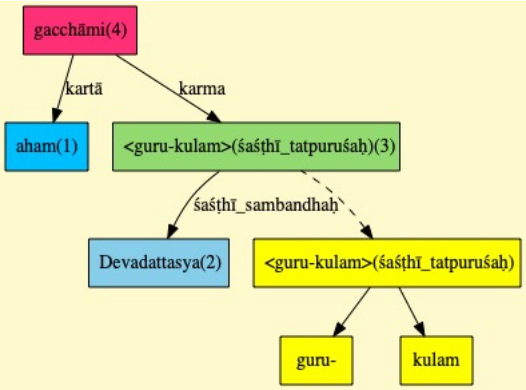
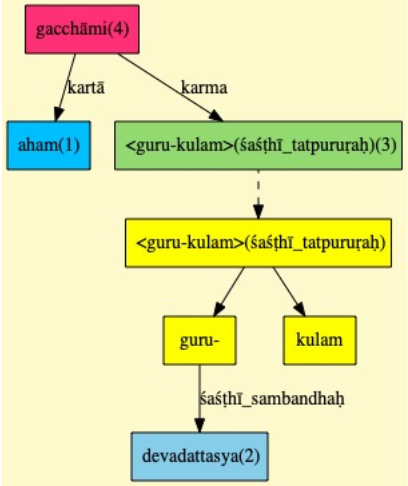


Illustration - 3



asaramrtha compound



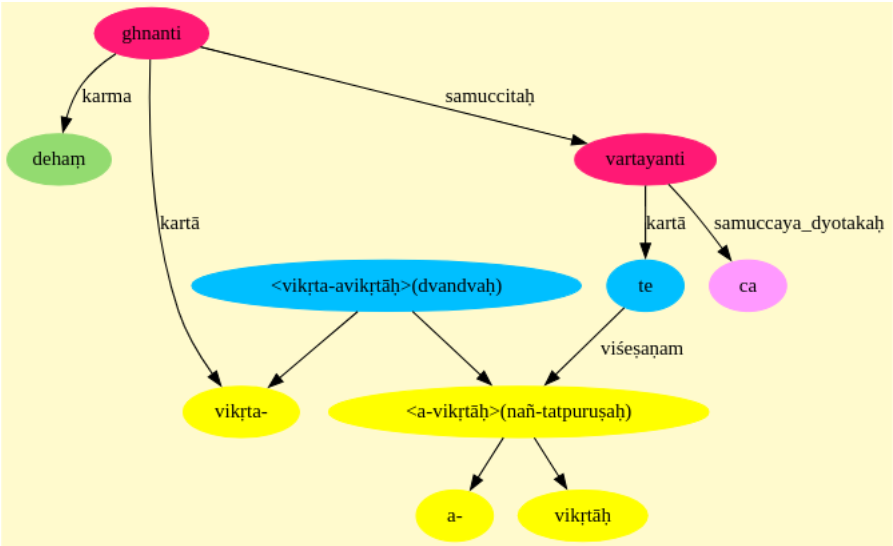
Demo

Challenges:

How to model Yogyatā?

How to model Sāmarthya?

More challenges



Websites

Some important websites

- https://sanskrit.inria.fr – Sanskrit Heritage Site; Segmentation
- https://sanskrit.uohyd.ac.in/scl – Saṁsādhanī University of Hyderabad Server; parser
- http://scl.samsaadhanii.in/scl – Saṁsādhanī Cloud server; parser
- https://start.samsaadhanii.in – Dashboard for Semi-automatic Annotation
- https://ereaders.samsaadhanii.in/books/ereaders – Annotated Books - E-readers

Dhanyavādaḥ