Pāṇini:An Information Scientist

Akshar Bharati
Amba Kulkarni
Department of Sanskrit Studies
University of Hyderabad
apksh@uohyd.ernet.in

Akshar Bharati is the personification of a group working on NLP with special emphasis to Indian languages giving due attention to the traditional Indian theories of grammar and language.

Circa 500 B.C.E.

Extant Grammar of the then prevalent Sanskrit Language

Around 4000 sūtras;

8 chapters 4 sections each

Aṣṭādhyāyī: 3 fold Importance

- The structure of Aṣṭādhyāyī: Arrangement of sūtras Programming Languages:

- An ‘exhaustive’ Grammar for Sanskrit,

- Theoretical Concepts useful for ‘analysing’ other languages.

Computational Linguists:
(Language: Means of coding the information.)
Information Coding: How much, Where and How

Claim:
Pāṇini was aware of the strength of language as
an information coding device.
And Pāṇini made the best use of this strength.

Evident from

- His style of presenting the information in sūtra
- The way he has analysed the Sanskrit Language

Kiparsky: Pāṇini used Brevity to achieve generalisation.

Maximum Use of anuvṛtti (factorisation)

Ram went home.

Ram ate an apple.

Ram went home and ate an apple.

- 1.3.2 upadeśe ac anunāsika it
- 1.3.3 hal antyam
- 1.3.4 na vibhaktau tusmā
- 1.3.5 ādi ñitudavāḥ
- 1.3.6 ṣaḥ pratyayasya
- 1.3.7 cuṭū
- 1.3.8 laśaku ataddhite

- 1.3.2 upadeśe (a)
  ac anunāsika(b) (=it(c)) abc
- 1.3.3 hal antyam(d)
- 1.3.4 na vibhaktau tusmā(e) (=it) adec
- 1.3.5 ādi (f) (=it)
- 1.3.5 ñitudavāḥ(g) (=it) afgc
- 1.3.6 ṣaḥ (h) pratyayasya(i) (=it) afhic
- 1.3.7 cuṭū(j) (=it) afijc
- 1.3.8 laśaku ataddhite(k) (=it) afikc

a{b + de + f[g + i (i + j + k)]}c

No Proper Nesting; Maṇḍūka pluti

- 6.1.84 ād(a) guṇaḥ(b)
- 6.1.85 vṛddhiḥ(c) eci(d) a
- 6.1.86 etyedhatyūṭsu (e) a c d
- 6.1.87 upasargāt(f) ṛti(g) dhātau(h) a c
- 6.1.88 vā supyāpiśale(i) f g h a c
- 6.1.89 autaḥ amśasoḥ(j)
- 6.1.90 eńi(k) pararūpaṃ(l) f h a

$$a\{b + c[d(1+e) + fh < g(1 + i)] + j + kl>\}$$

How are the complete phrases reconstructed?

Maximum advantage of features of Natural Language:

ākāṅkṣā (Expectancy): Major role in deciding the anuvṛtti

Example of borrowing from as many as 11 stras

Original sūtra:
3-3-65 kvaṇaḥ vīṇāyā ca

After anuvṛtti: 3-3-65: kvaṇaḥ vīṇāyā ca pratyayaḥ paraḥ ca ādyudāttaḥ ca dhātoḥ kṛt kriyāyām kriyārthāyām bhāve akartari ca kārake sajñāyām ap upasarge vā nau (anuvṛtti from 11 different stras)

Some Statistics:

Total sūtras: (3984) 4000

Total Words (with sandhi): (7007) 7000

Total Sandhi split words: 9843

Total words after repeating the words with anuvṛtti: 40,000

Compression because of anuvṛtti: 1/6

In terms of byte size, compression is 1/3.

Normal Arrangement of Alphabet

a ā i ī u ū ṛ ḷ e ai o au ṃ ḥ

k kh g gh ń

c ch j jh ñ

ṭ ṭh ḍ ḍh ṇ

t th d dh n

p ph b bh m

y r l v

ś ṣ s h

Pāṇini required several(42) subsets of this alphabet to describe various operations.

It is not advisable to give 42 names to these sets.
It will be difficult to memorize the association.

These are Partially ordered sets.

Pāṇini arranged them linearly in the form of 14 ṣivasūtras.

a i u N

ṛ ḷ K

e o c

ai au Ń

h y v r T

l N

ñ m ń ṇ n M

jh bh Ñ

gh ḍh dh Ṣ

j b g ḍ d S

kh ph ch ṭh th c ṭ t V

k p Y

ś ṣ s R

h L

Optimality of these sūtras is
proved independently by

Kiparsky (linguistically)
and Petersen (mathematically)

Given a set of Partially Ordered sets,
Now it is possible to tell
Whether the elements are
Shivasutra encodable or not.
Ref: Petersen(2008)

Māheśvarasūtra

a i u Ṇ

ṛ ḷ K

e o Ṅ

ai ao C

h y v r T

l Ṇ

$a\d{n}$ == > {a i u } ; $a\d{n}$ == > {a i u ṛ ḷ e o ai ao h y v r t l}

$i\d{n}$ == > {i u } ; $i\d{n}$ == > {i u ṛ ḷ e o ai ao h y v r t l}

5 sūtras with aṆ

ढ्र लोपे पूर्वस्य दीर्घः अणः 6.3.110

के अणः (अङ्गस्य ह्रस्वः) 7.4.13

अणः अप्रगृह्यस्य अनुनासिकः (वा) 8.4.56

उरणः रपरः 1.1.50

अणुदित् सवर्णस्य अप्रत्ययः 1.1.68

सामर्थ्य (Ability to convey proper meaning)

ढ्र लोपे पूर्वस्य दीर्घः अणः 6.3.110
के अणः (अङ्गस्य ह्रस्वः) 7.4.13
अणः अ -प्रगृह्यस्य अनुनासिकः (वा) 8.4.56

ह्रस्व and दीर्घ properties of a vowel.
Only Vowels can get प्रगृह्य संज्ञा

प्रसिद्धि (Frequency of usage)

उरणः रपरः 1.1.50

No example involving members of bigger set.

- The effect of the rule is nullified by other sūtra, OR
- The application of sūtra leads to undesirable redundancy in some other sūtra

लिङ्ग (indicator/marker)

अणुदित् सवर्णस्य च अप्रत्ययः

उः ऋत् (== > तपर)

तपरः तत्कालस्य (सवर्णस्य)

== > sūtra is applicable for 'ऋ'

and ऋ ϵ aṆ$_2$

== > ण् is the second ण्

$i\d{n} ==> \{\text{i u }\};$

$i\d{n} ==> \{\text{i u ṛ ḷ e o ai ao h y v r t l}\}$

लाघव (economy)

इ उ $==>$ य्व

इणः $==>$ य्वोः

1+.5+1+.5(=3) .5+.5+2+.5 (=3.5)

व्याख्यानतः विशेष प्रतिपत्तिः न हि सन्देहात् अलक्षणम्

Had Pāṇini used some other consonant as an anubandha, he would have lost an opportunity to train the students in paying attention to the different means of information coding a language employs.

Should we then not conclude that
Pāṇini was aware of ambiguities a natural language has and wanted to train the students of vyākaraṇa to pay attention to different sources of information available for disambiguation?
And that he uses the very first opportunity to train the students – right from the Māheṣvarasūtras with which the study of Aṣṭādhyāyī commences?

For a person working in NLP
the following questions are important

- Where does the language code information?
- How much information does it code?
- How does the language code information?

Dynamics of Information coding in Sanskrit

'Where' is the information coded?

रामः ग्रामम् गच्छति

रामेण ग्रामः गम्यते

1$^{\text{st}}$ reaction:

If kartari prayoga(active voice)

- kartā – > Nominative Case
- karma – > Accusative Case

If karmaṇi prayoga(passive voice)

- kartā – > Instrumental Case
- karma – > Nominative Case

It is also necessary to

- state noun-verb agreement
- account for pro-drop as in gacchāmi

- लः कर्मणि च भावे च अकर्मकेभ्याः (कर्तरि) 3.4.69
- अनभिहिते 3.1.1
- कर्तृकरणयोः तृतीया 2.3.18
- कर्मणी द्वितीया 2.3.2
- प्रातिपदिकार्थलिङ्गपरिमाणवचनमात्रे प्रथमा 2.3.46

"How much" information is coded?

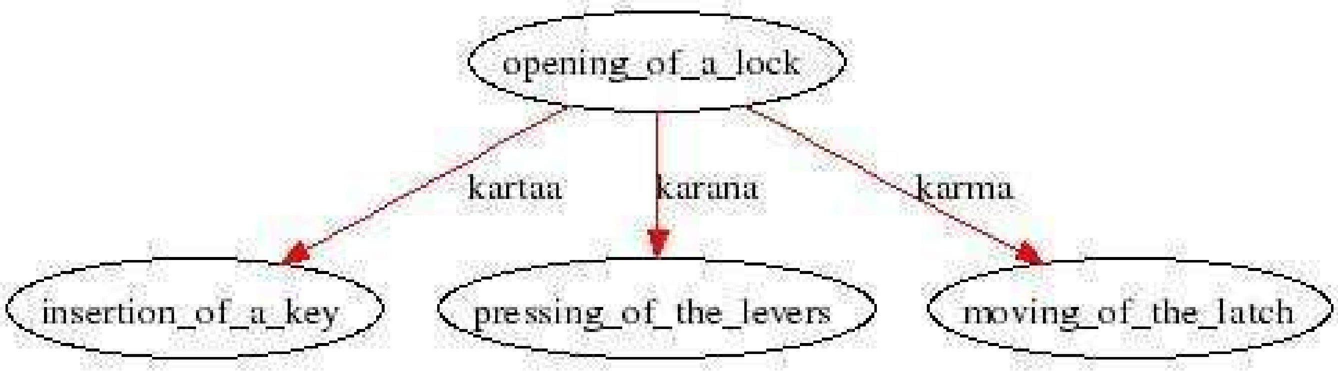
1. रामः कुञ्चिकया तालम् उद्घाटयति
2. कुञ्चिका तालम् उद्घाटयति
3. तालः उद्घाट्यते

रामः कुञ्चिका तालः == > कर्ता

राम == > Agent

कुञ्चिका == > Instrument

तालः == > Goal / Patient



स्वतन्त्रः कर्ताः

Greatness of *Pāṇini* lies in
identifying EXACTLY HOW MUCH
information is coded in a language string.

$===>$
Upper Bound for the possible Analysis
using only a language string and grammar.

We can extract only that which is available in the language string ‘without any requirement of additional knowledge’.

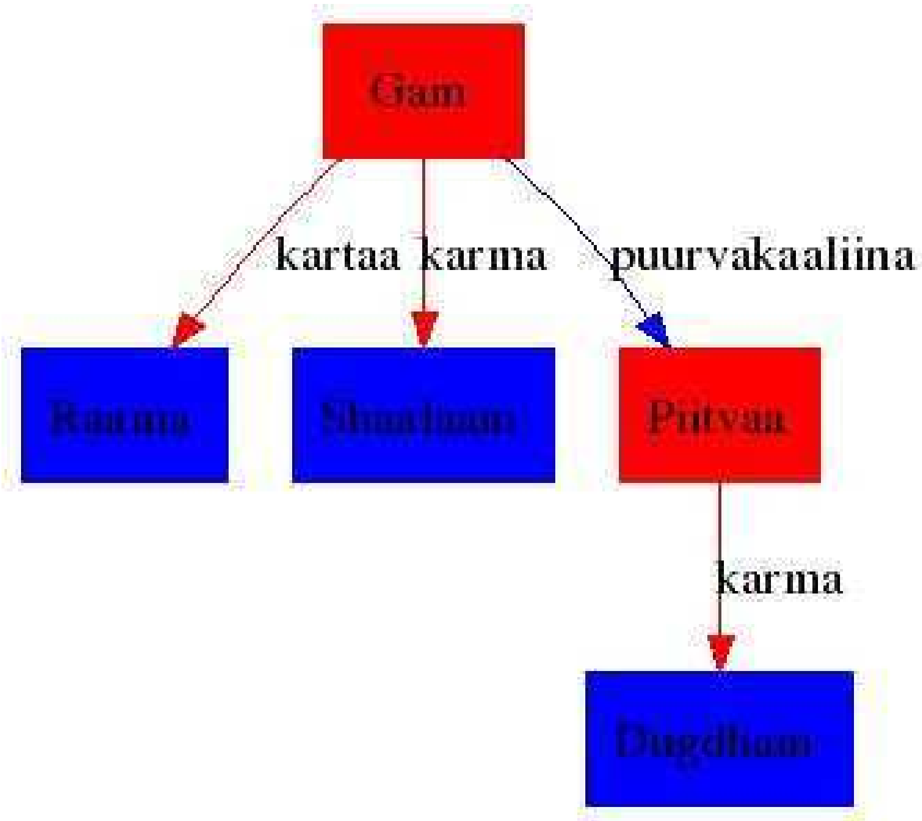
Analogy:
We can not do high quality work with low quality energy.

How is the information Coded?

रामः दुग्धम् पीत्वा शालाम् गच्छति

2 verbs with 2 expectancies each, and only 3 nouns!

रामः दुग्धम् पीत्वा शालाम् गच्छति



Who drank milk?

समानकर्तृकयोः पूर्वकाले

Information is coded as a “Language Convention”
== > Different Languages may have different conventions.

== > Automatic Translation may lead to Ungrammatical Sentences

Mohan dropped the Melon and Burst.

वनात् ग्रामम् अद्य उपेत्य ओदनम् आश्वपत्येन अपाचि.

Where does the language code information?

How much information does it code?

How does a language code information?

Claim: Any grammar that is developed with these questions in mind will be a grammar truly in *Pāṇinian* Spirit.

Structure of Aṣṭādhyāyī:

A programmer's Perspective

Important issues:

- How are the rules ordered?
- If more than one rules are applicable then how is the conflict resolved?

Data Encapsulation:

- All the indicators trigger some functions.
- The *ñi*

indicator indicates that such a root takes the suffix *kta* in the sense of present tense, as in

ñidhṛṣā + *kta* - > *dhṛṣṭa*

Subroutines: Marking *anubandha*s

- *upadeśe ac anunāsika it 1.3.2*
- *hal antyam 1.3.3*
- *na vibhaktau tusmā 1.3.4*
- *ādiḥ ñiṭudavaḥ 1.3.5*
- *ṣaḥ pratyayasya 1.3.6*
- *cutū 1.3.7*
- *laśaku ataddhite 1.3.8*
- *tasya lopaḥ 1.3.9*

If we take into account the '*anuvrutti*', the rules may be rewritten as

- *upadeśe*
  - *ac anunāsika (=)it 1.3.2*
  - *hal antyam 1.3.3*
    - *na vibhaktau tusmā (=it)1.3.4*
  - *ādiḥ*
    - *ñiṭudavaḥ (=it) 1.3.5*
    - *pratyayasya*
      - *ṣaḥ (=it) 1.3.6*
      - *cutū (=it) 1.3.7*
      - *laśaku (=it) ataddhite 1.3.8*

The parallel between the *Pāṇini*'s *sūtra*s and the computer algortihm

```
if(the input is from UPADE.SA)
    Mark the ANUNASIKA AC as INDICATOR
  if(the last var.na is HAL)
     if(it is neither VIBAKTI nor TUSMA) { Mark it as INDICATOR}
  if(the INITIAL var.na is either ~ni or tu or du){ Mark it as INDICATOR}
  if(the INITIAL var.na is a PRATYAYA) {
     if(it is .sa {MARK it as INDICATOR}
     if(it is from ca_varga or ta_varga) {Mark it as INDICATOR}
     if(it is NOT TADDHITA)
        if(it is either la or `sa or ka_varga){Mark it as INDICATOR}
  }
```

The operations in *Pāṇini*'s grammar:

- assigning a name
- substitution
- insertion
- deletion

Contd...

Ingenious usage of *vibhaktis*:

A typical context sensitive rule

$$\alpha\beta\gamma- > \alpha\delta\gamma$$

- 5^{th} case to indicate the left context
- 7^{th} case to indicate the right context
- 6^{th} case to indicate which element will undergo a change and
- 1^{st} to indicate what it will change to.

$$5\ 6\ 7 == > 5\ 1\ 7$$

An example from *Aṣṭādhyāī*.

ataḥ roḥ aplutāt aplute (ut ati saṃhitāyām) 6.1.113

ataḥ{5} ru{6} aplut{5} apult{7} (ut{1} at{7})

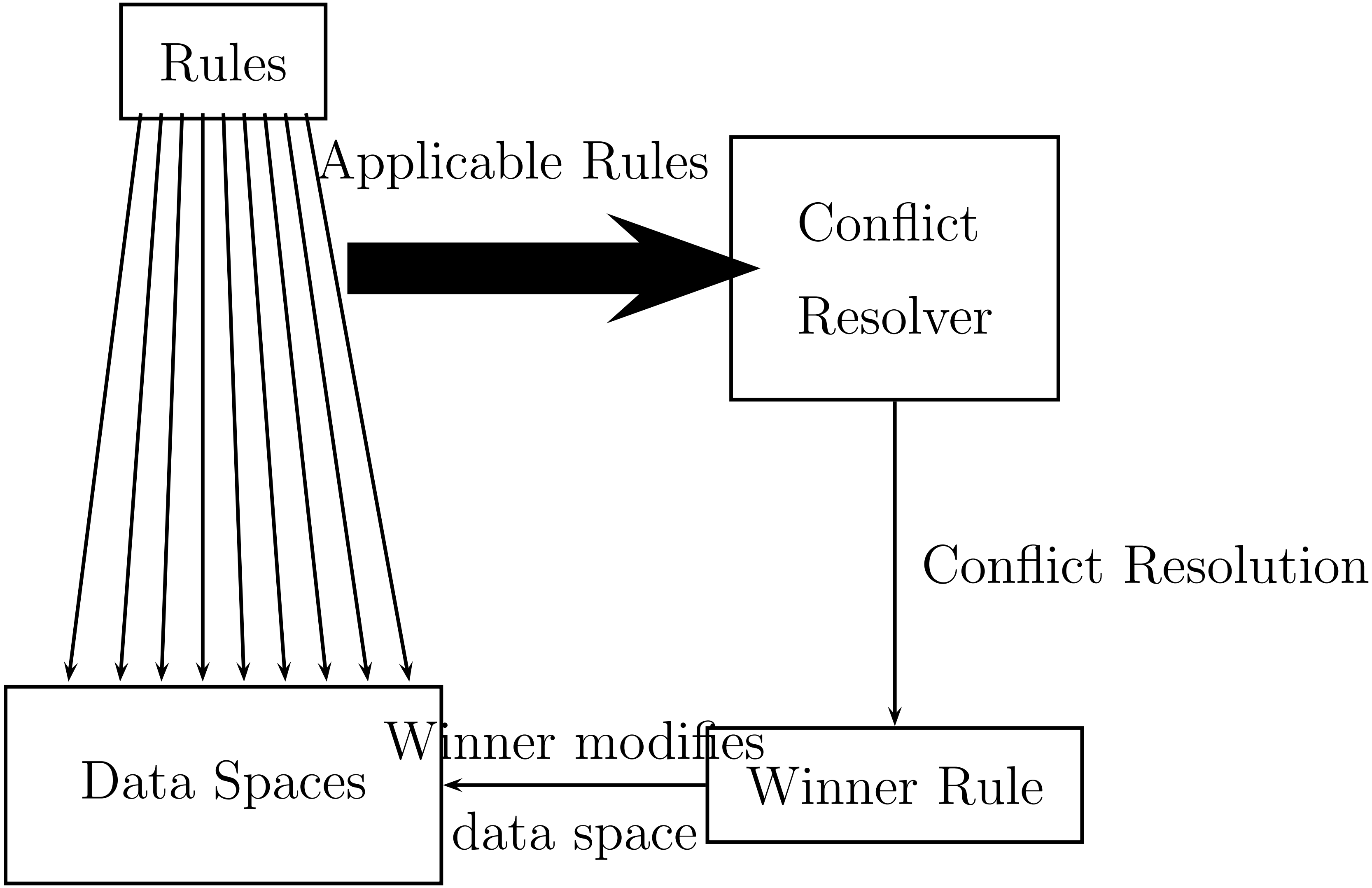
apluta_at ru apluta_at –> apluta_at ut apluta_at

‘siva ru arcya –> s̀iva u arcya

How are the rules triggered?
Typical Grammarian's view:

The rules in the *sapāda saptādhyāyī* seek for an opportunity(*nimitta*) to act on an input. In case there is a conflict, there are certain conflict resolution techniques (described as *paribhāṣā*), which come into play.The *nimitta* is in the form of a context. When no other rules are applicable, then the rules in tripādī are applied sequentially thereby ending the process of derivation.

Event Driven Programming?


Rules
Applicable Rules
Conflict
Resolver
Conflict Resolution
Data Spaces
Winner modifies
data space
Winner Rule

Ordering of Rules...

Asiddhavat, Asiddha and Asiddham

- *pūrvatra* ***asiddham*** *8.2.1*
- ***asiddhavat*** *atra ābhāt 6.4.22*
- *ṣtvatukor**asiddhaḥ** 6.1.86*

Contd...

Asiddham- *pūrvatra* ***asiddham*** *8.2.1*

- The output of the rules in this part is not available to the earlier rules. ⇒ *tripādī* should follow *sapāda saptādhyāyī*

- The rule makes the output of each of the following *sūtra* unavailable to the previous rules. ⇒ The rules within *tripādī* should be followed lineraly.

- Whole *tripādi* may be considered to be a single subroutine

Asiddhavat- asiddhavat *atra ābhāt 6.4.22*

Consider the derivation: *s̀ādhi* from *s̀ās + hi*.

Applicable *sūtras*:

- *hujhulbhyo herdhi 6.4.101*
- *s̀ā hau 6.4.35*

6.4.101: s̀ās + hi -> s̀ās + dhi

6.4.35: s̀ās + hi -> s̀ā + hi

Contd...

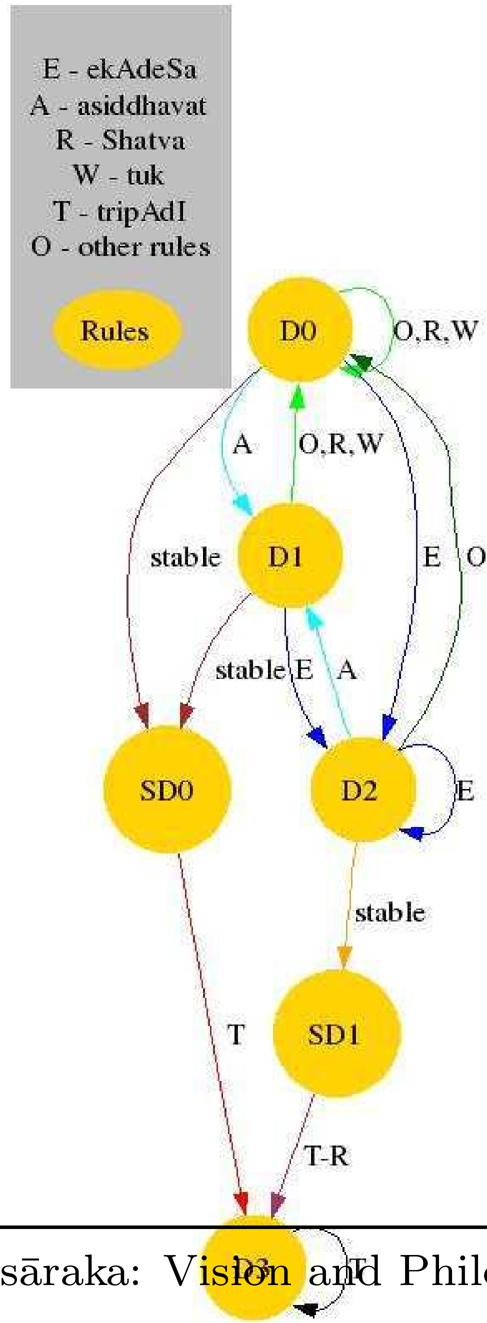
- Task parallelism is achieved.
  - If 6.4.101 is applied, then the conditions for applying 6.4.35 are not met.
  - If 6.4.35 is applied first, then the conditions for 6.4.101 would not be met.
  - *ṡās + hi* -> *ṡā + hi*: Result invisible to 6.4.35.
- Economy is achieved.
  - R: a b - > c d factored as:

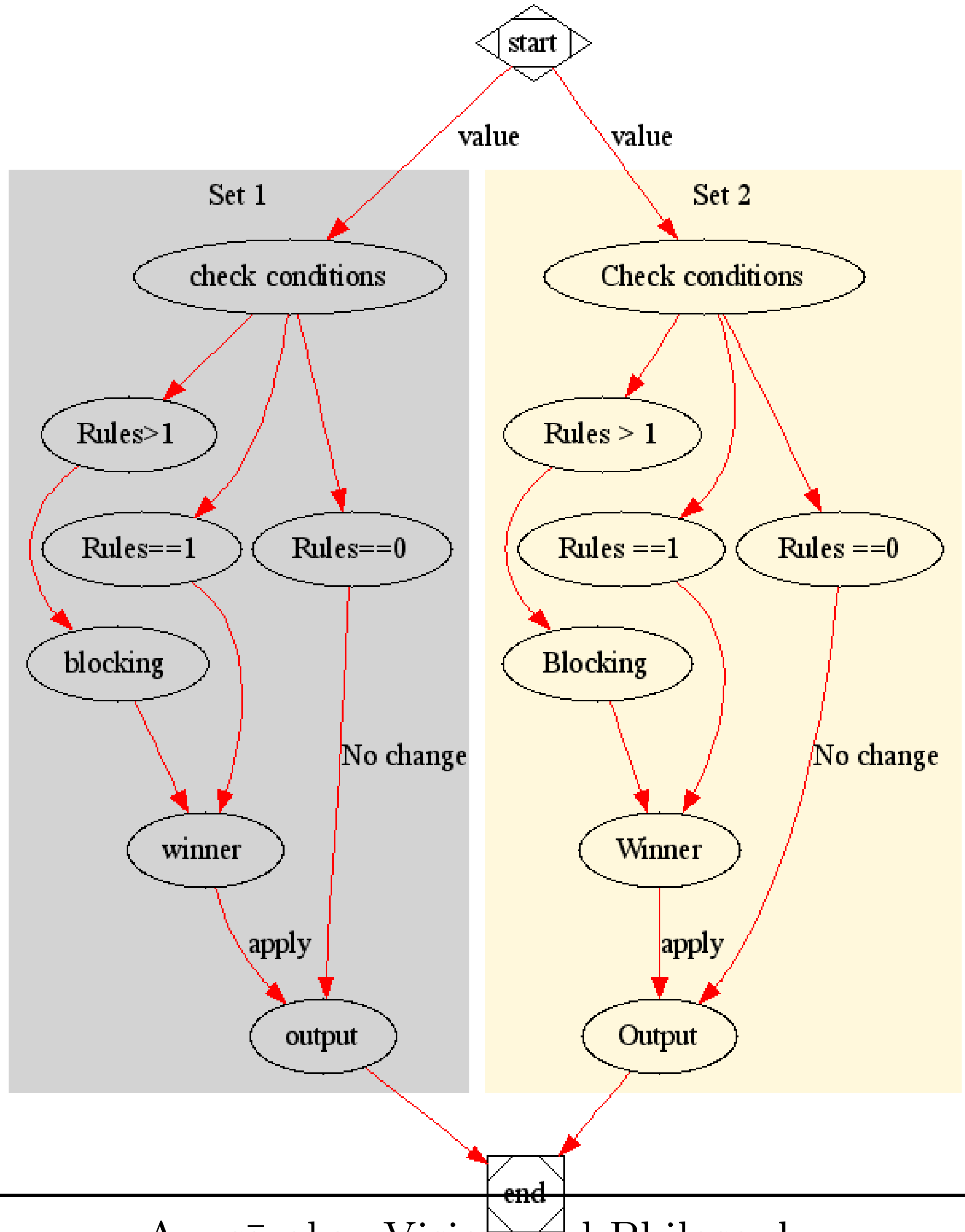
$$R_1 : ab- > cb, R_2 : ab- > ad$$

  - $n_1 + n_2$ rules instead of $n_1 * n_2$ rules.

Asiddhaḥ- *ṣtvatukor**asiddhaḥ** 6.1.86*

'ekādeṡa' section is invisible to the 'ṣatva' and 'tuk' processes.


E - ekAdeSa
A - asiddhavat
R - Shatva
W - tuk
T - tripAdI
O - other rules
Rules
D0
O,R,W
A
O,R,W
D1
stable
E
O
stable
E
A
SD0
D2
E
stable
T
SD1
T-R


start
value
value
Set 1
check conditions
Rules>1
Rules==1
Rules==0
blocking
No change
winner
apply
output
Set 2
Check conditions
Rules > 1
Rules ==1
Rules ==0
Blocking
No change
Winner
apply
Output
end