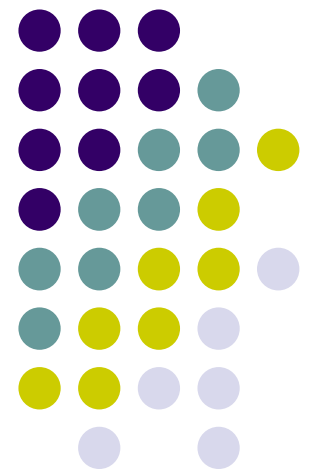
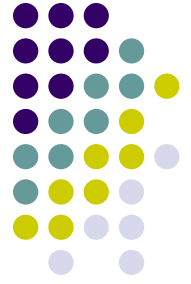


A Two-Stage Constraint Based Dependency Parser for Free Word Order Languages

Samar Husain

LTRC, IIT Hyderabad,
India





Introduction

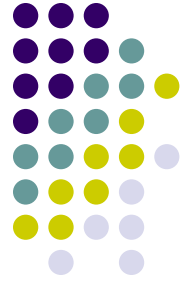
- Broad coverage parser
 - Very crucial
 - IL-IL MT systems, IE, co-reference resolution, etc.
- Attempt to make an hybrid parser



Levels of Language Analysis

- Morphological analysis (Morph Info.)
- Analysis in local context (POS tagging)
- Sentence analysis (Chunking, Parsing)

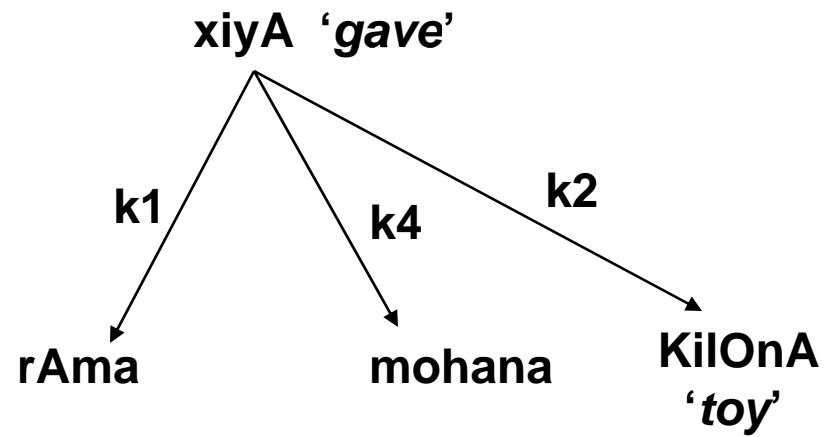
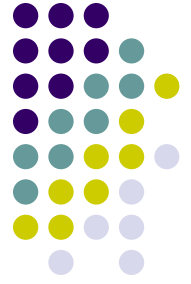
- Semantic analysis (Word sense disambiguation, etc.)
- Discourse processing (Anaphora resolution, Informational Structure, etc.)

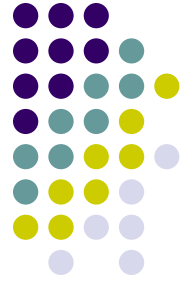


Example

- rAma ne mohana ko KilOnA xiyA |
Ram 'ERG' Mohana 'DAT' book gave
'Ram gave a book to Mohan'

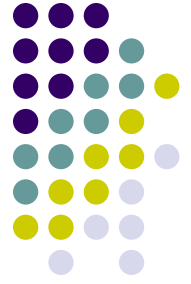
Example – Parsed Output





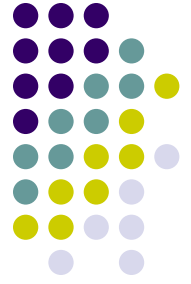
The design

- Constraint satisfaction problem
 - Inviolable constraints
 - Rule based
 - Violable constraints
- Selective resolution of demands
- Repair
- Partial Parses



Inviolable constraints

- Structural constraints
 - Dependency tree structure
- Verbal constraints
 - Demand frames (and transformation)
- Other non-verbal lexical constraints



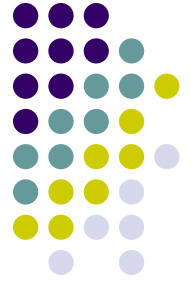
Implemented Parser

- Two stage strategy
 - Appropriate constraints formed
- Stage I (Intra-clausal relations)
 - Dependency relations marked
 - Relations such as k_1 , k_2 , k_3 , etc. for each verb
- Stage II (Inter-clausal relations & conjunct relations)
 - Conjuncts, relative clauses, complex verbs, etc



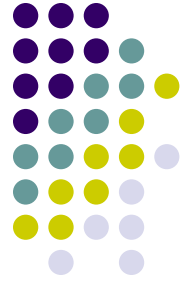
Demand Frame for Verb

- A demand frame or karaka frame for a verb indicates the demands the verb makes
- It depends on the verbal semantics and the tense, aspect and modality (TAM) label.
- A mapping is specified between karaka relations and vibhaktis (post-positions, suffix).



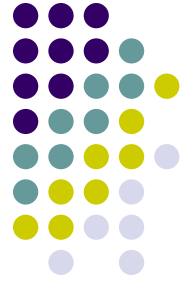
Demand Frame

- It specifies what karakas are mandatory or optional for the verb and what vibhaktis (post-positions) it takes
- Each verb belongs to a specific verb class
 - Each class has a basic karaka frame
- Each TAM specifies a transformation rule



Transformations

- Basic frame
 - *rAma ø mohana ko KilOnA xiwA hE* |
 - Ram *gives* Mohan a book
- Transform basic frame based on the TAM
 - *rAma ne mohana ko KilOnA xiyA* |
 - Ram *gave* Mohan a book
 - *rAma ko mohana ko KilOnA xenA padZA* |
 - Ram *had to give* Mohan a book
 - Appropriate transformation applied



Example

- rAma ne mohana ko KilOnA xiyA |
Ram 'ERG' Mohana 'DAT' book gave
'Ram gave a book to Mohan'

Karaka Frame: xe [_wA hE] (give)

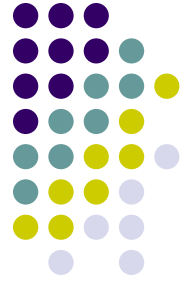


arc-label	necessity	vibhakti	lextype	src-pos	arc-dir
k1	m	0	n	l	c
k2	m	0 ko	n	l	c
k3	d	se	n	l	c
k4	d	ko	n	l	c

Transformation Rule – yA (TAM)



arc-label	necessity	vibhakti	lextype	src-pos	arc-dir
k1	m	ne	-	-	-



Karaka Frame

yA TAM

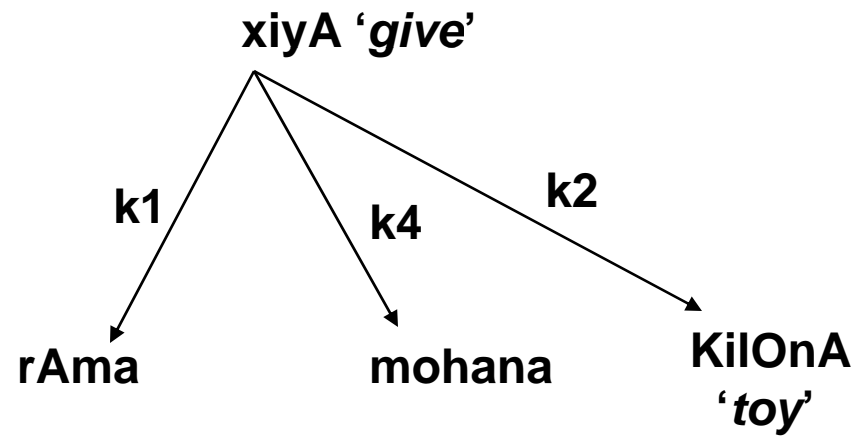
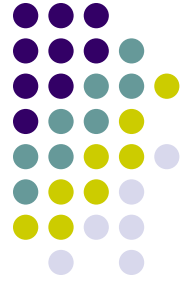
rAma ne mohana ko KilOnA xiyA |

Transformed frame for *xe* after applying the *yA* transformation

arc-label	necessity	vibhakti	lextype	src-pos	arc-dir
k1	m	ne	n		c
k2	m	0 ko	n		c
k3	d	se	n		c
k4	d	ko	n		c

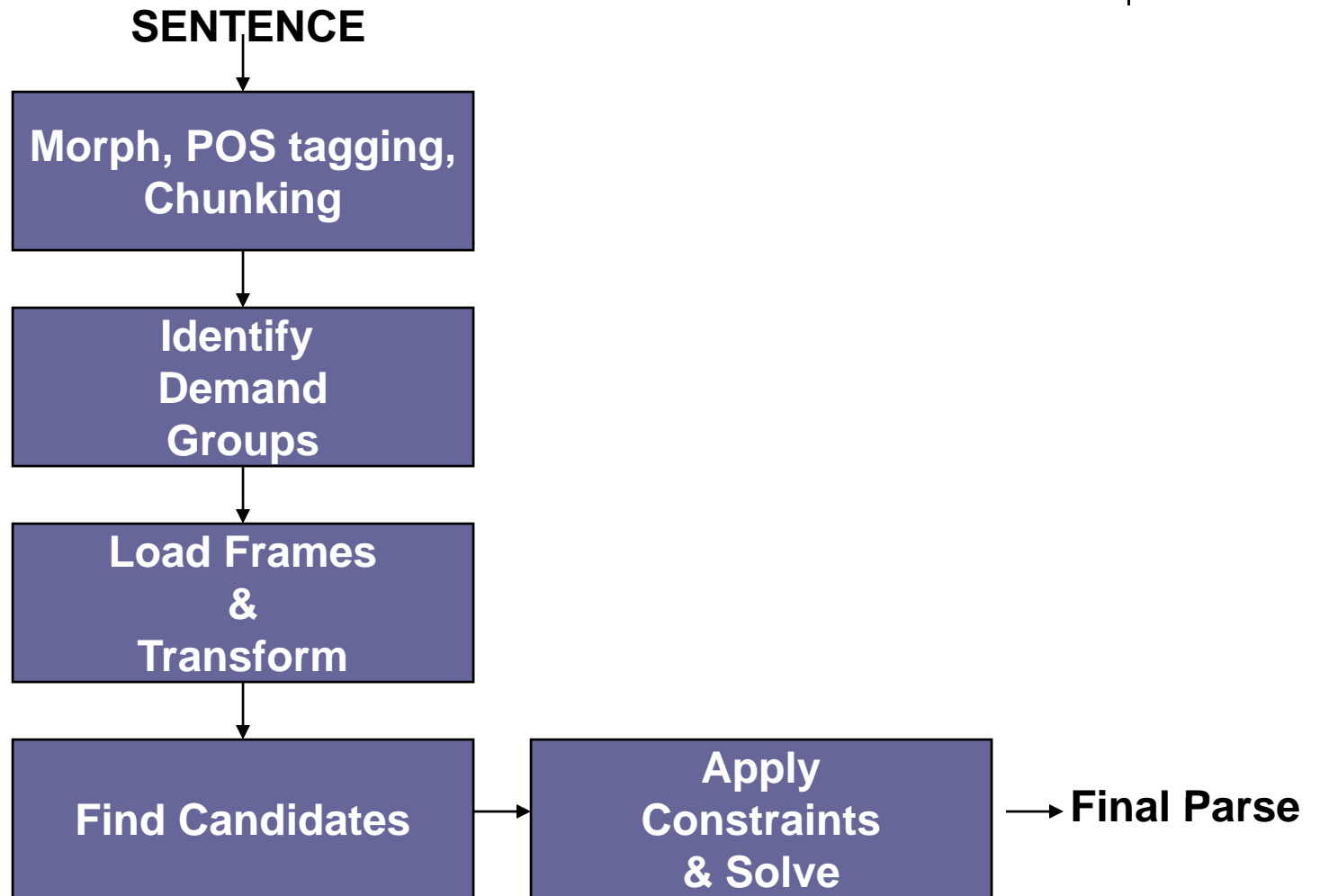
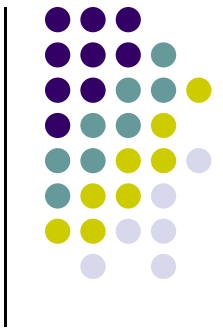
0 → ne

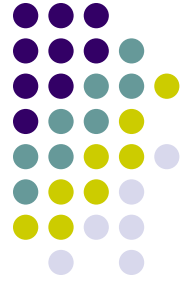
Parsed Output



- **rAma ne mohana ko KilOnA xiyA |**
Ram 'ERG' Mohana 'DAT' book gave
'Ram gave a book to Mohan'

Steps in Parsing

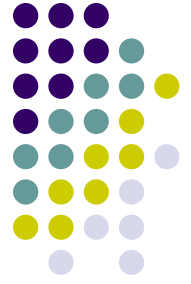




Example:

- rAma ne mohana ko KilOnA xiyA |
Ram 'ERG' Mohana 'DAT' book gave
'Ram gave a book to Mohan'

Identify the demand group, Load and Transform DF

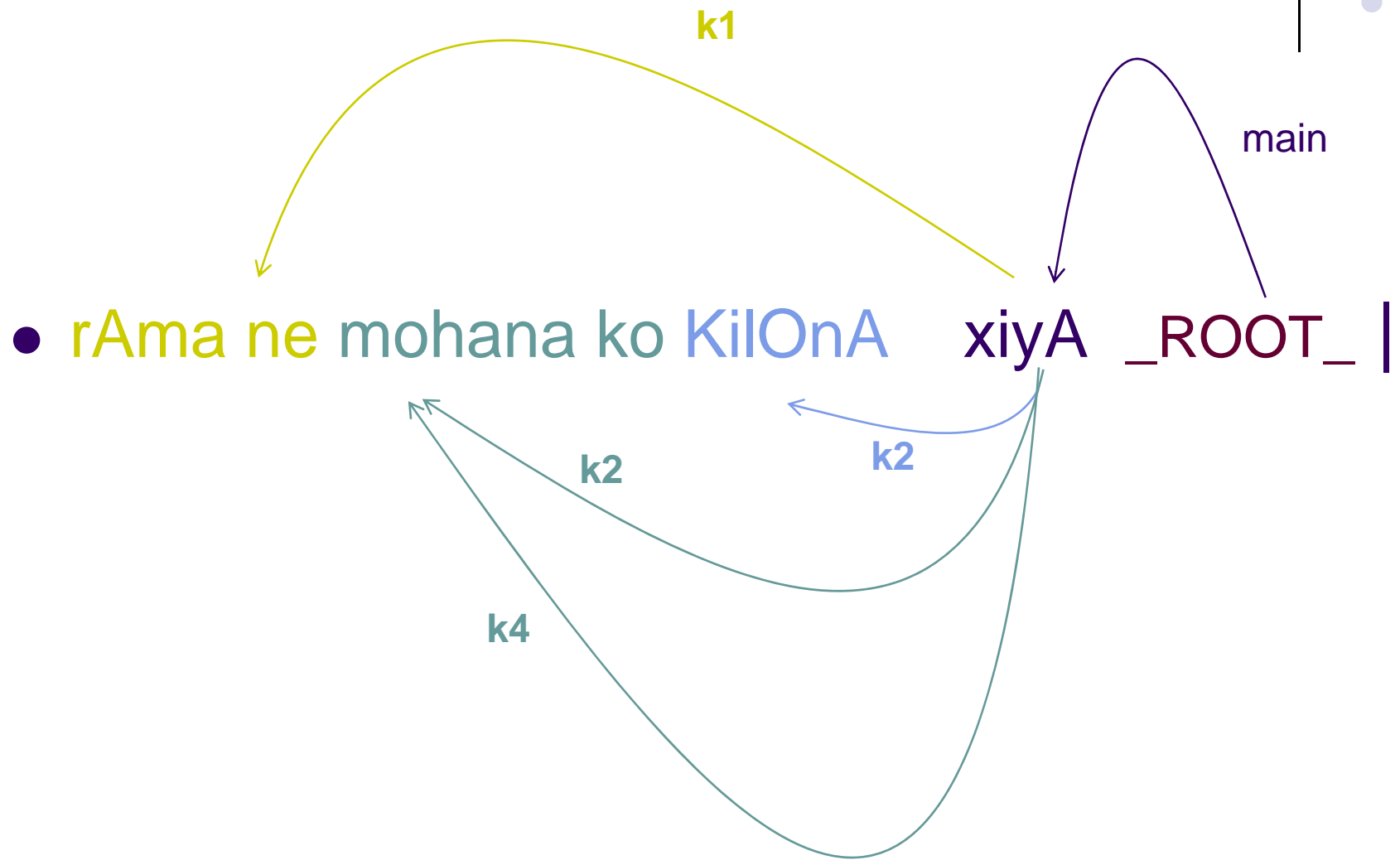


- xiyA
 - The only verb
- Transformed frame
 - Use 'yA' TAM info.

arc-label	necessity	vibhakti	lextype	src-pos	arc-dir
k1	m	ne	n		c
k2	m	0 ko	n		c
k3	d	se	n		c
k4	d	ko	n		c



Candidates





Structural constraints

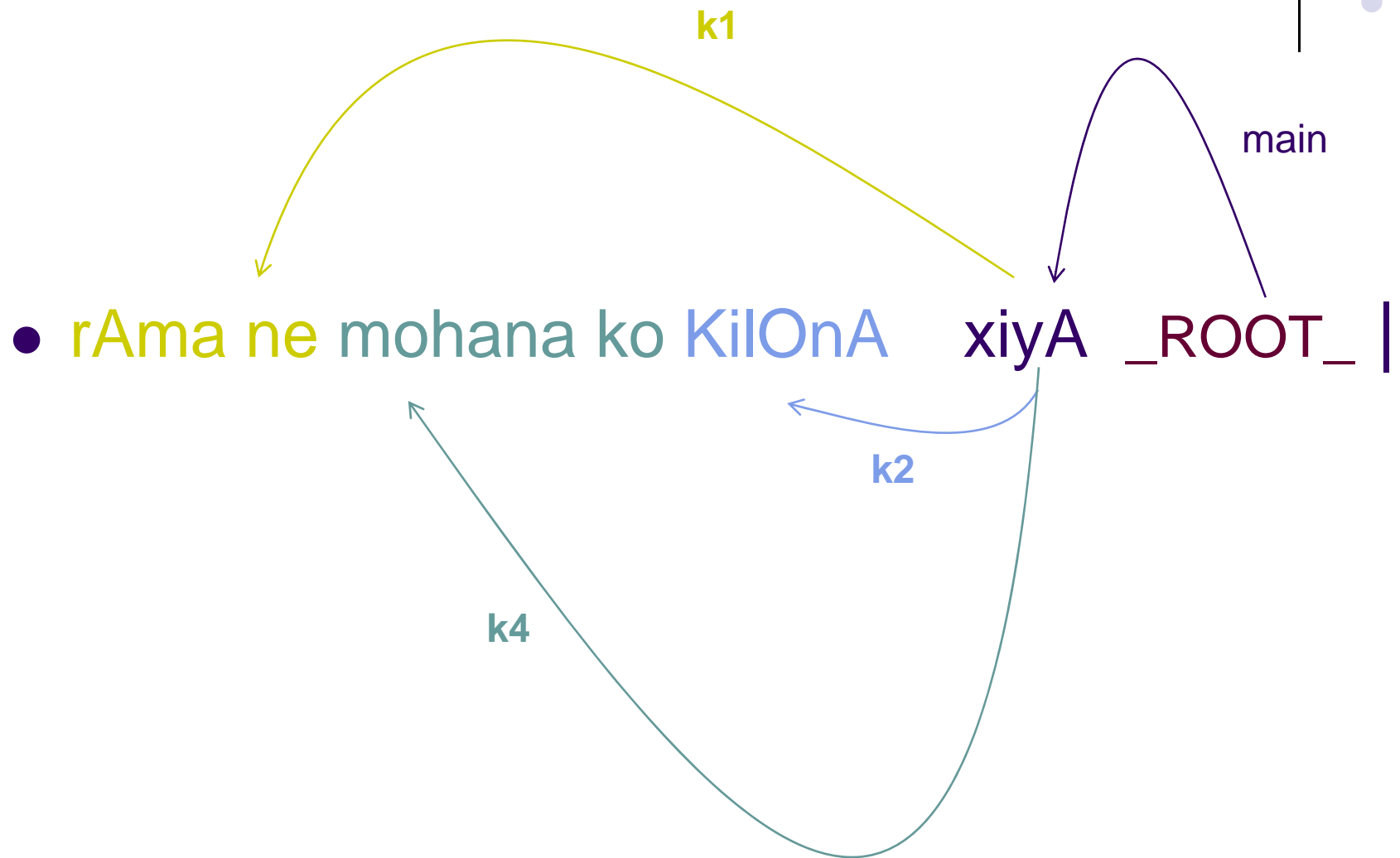
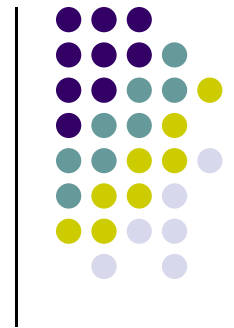
- **C1:** For each of the **mandatory demands** in a demand frame for each demand group, there should be **exactly one outgoing edge** labeled by the demand from the demand group.
- **C2:** For each of the **optional demands** in a demand frame for each demand group, there should be **at most one outgoing edge** labeled by the demand from the demand group.
- **C3:** There should be **exactly one incoming arc** into **each source group**.



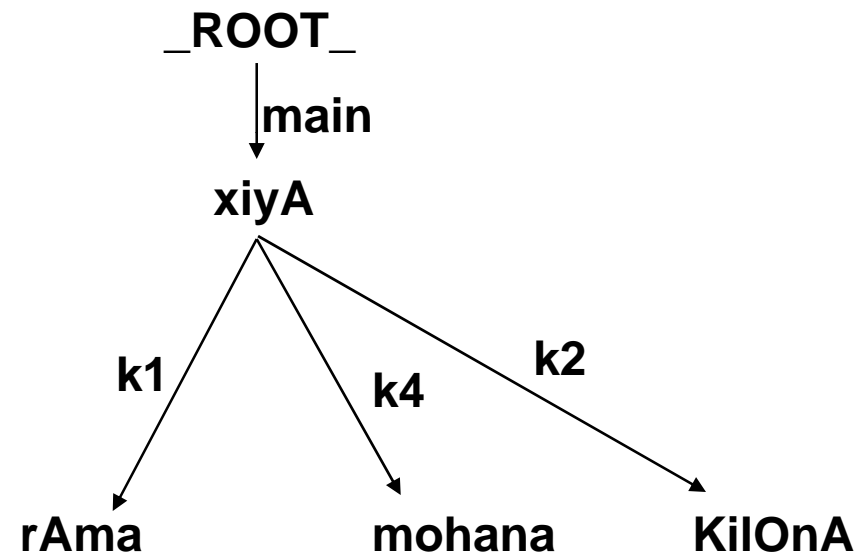
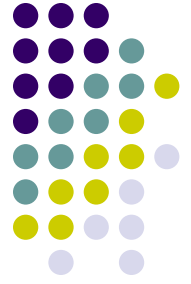
Inviolable constraints

- A parse of a sentence is obtained by satisfying all the above constraints
- Ambiguous sentences have multiple parses
- Ill formed sentences have no parse.

Parse - I

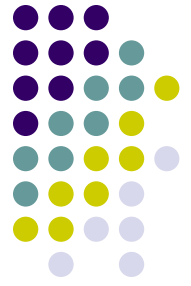


Parse - I



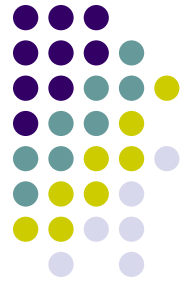
- **rAma ne mohana ko KilOnA xiyA |**
Ram 'ERG' Mohana 'DAT' book gave
'Ram gave a book to Mohan'

Integer Programming Constraints



- X_{ijk} represents a possible arc from word group i to j with karaka label k
- It takes a value **1** if the solution has that arc and **0** otherwise. It cannot take any other values.
- The constraint rules are formulated into constraint equations.

Constraint Equations



C1: For each demand group i , for each of its mandatory demands k , the following equalities must hold:

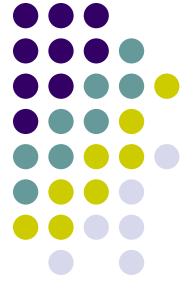
$$M_{ik} : \sum_j X_{ikj} = 1$$

C2: For each demand group i , for each of its optional or desirable demands k , the following inequalities must hold:

$$O_{ik} : \sum_j X_{ikj} \leq 1$$

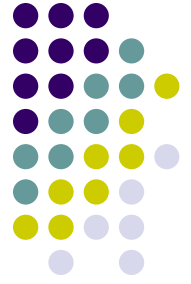
C3: For each of the source groups j , the following equalities must hold:

$$S_j : \sum_{ik} X_{ikj} = 1$$



Multiple Frames

- If more than one karaka frame for a verb
 - Call Integer Programming package for each frame
- If more than one demand groups (e.g., multiple verbs) in the sentence with multiple demand frames
 - Call Integer Programming package for each combination of such frames



Other frames

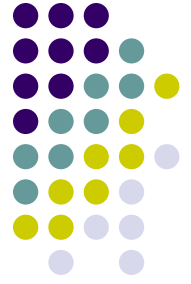
- Common karaka frame
 - Attached to each karaka frame
 - Preference given to main frame if there are clashes
- Fallback karaka frame
 - required karaka frame is missing
 - Graceful degradation

Stage I: Types being handled



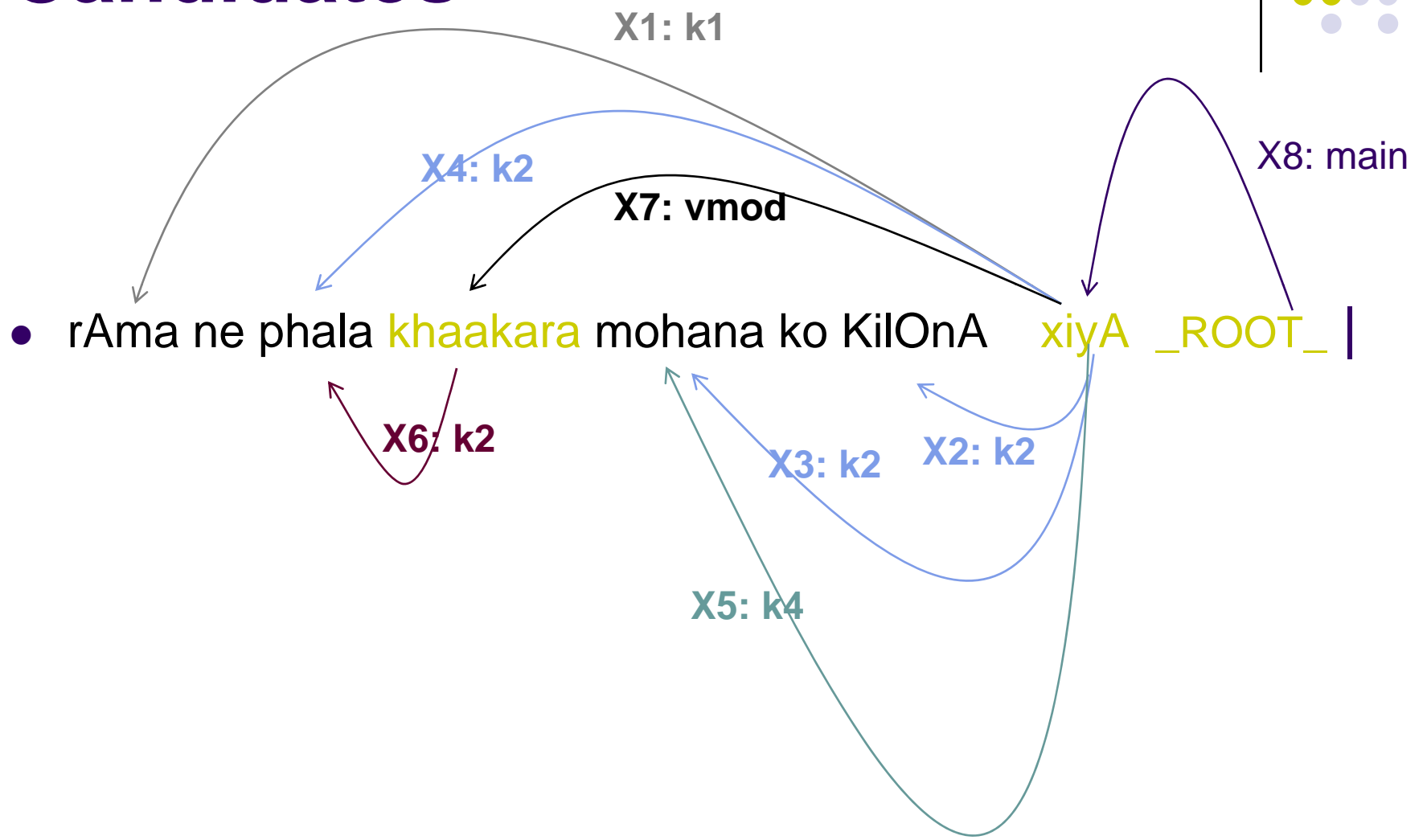
- Simple Verbs
- Non-finite verbs
 - wA_huA
 - wA_hI
 - nA
 - kara
 - 0_rahe, etc.
- Copula
- Genitive

Example (Complex Sentence)



- rAma ne phala khaakara mohana ko
Ram 'ERG' fruit 'having eaten' Mohan 'DAT'
KilOnA xiyA
toy gave
'Having eaten the fruit Ram gave the toy to Mohan'

Candidates

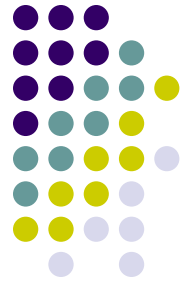




Constraint Equations

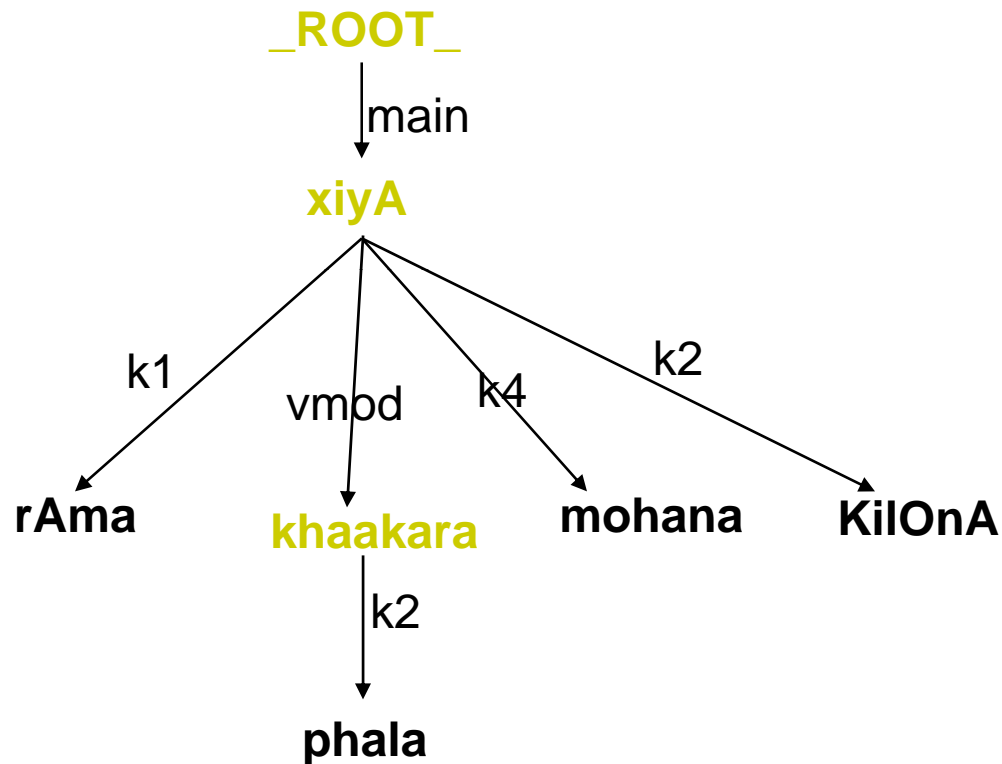
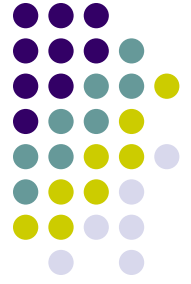
- Verb 'xe'
 - Mandatory Demands (C1)
 - $k1 \rightarrow x1 = 1$
 - $k2 \rightarrow x2 + x3 + x4 = 1$
 - Optional Demands (C2)
 - $k4 \rightarrow x5 \leq 1$
- Verb 'khaa'
 - Mandatory Demands (C1)
 - $k2 \rightarrow x6 = 1$
 - $vmod \rightarrow x7 = 1$
- `_ROOT_`
 - C1
 - $Main \rightarrow x8 = 1$

Constraint Equations (contd.)



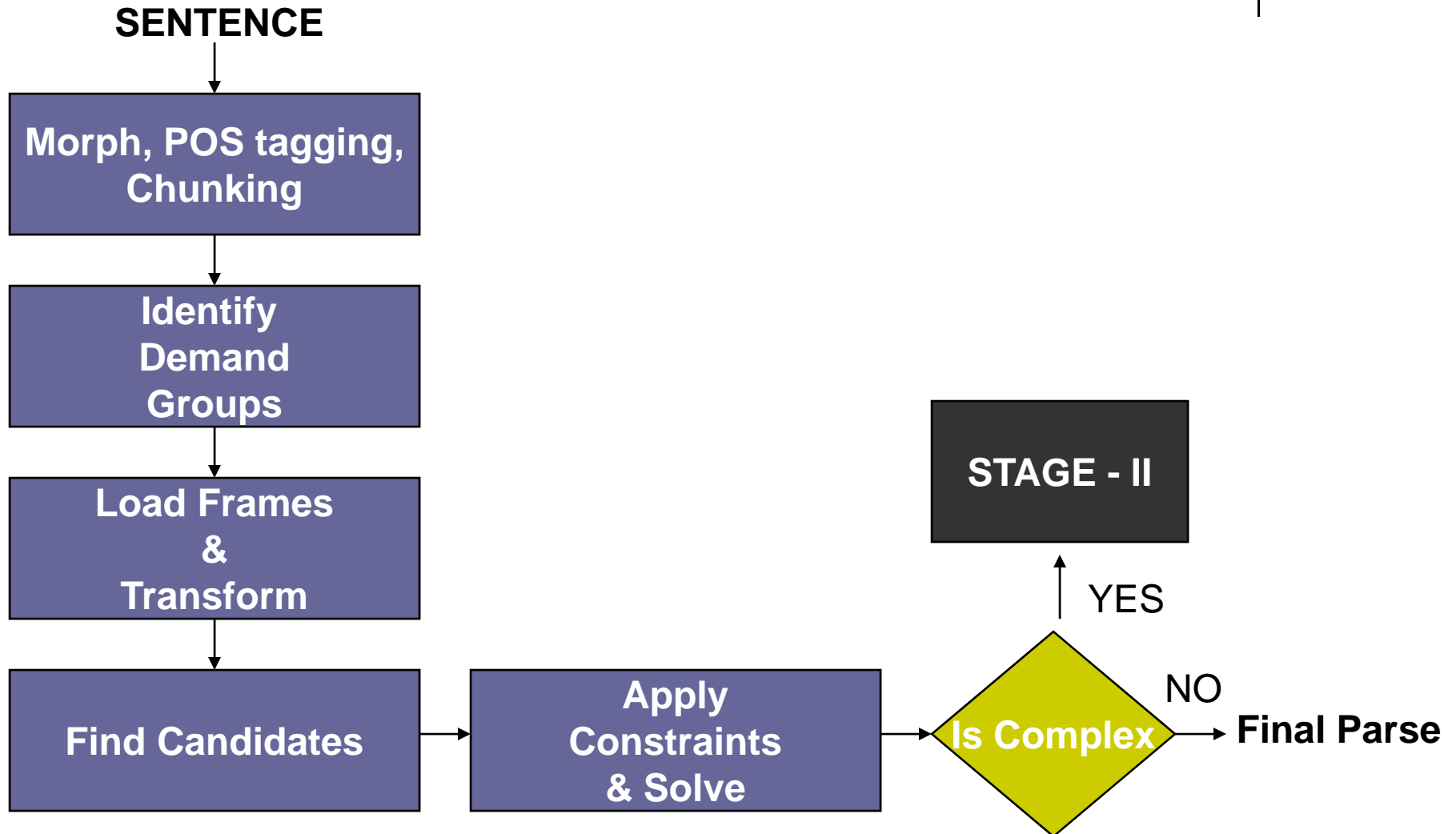
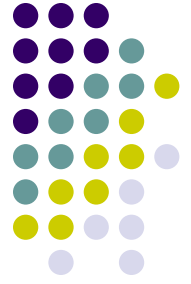
- Incoming Arcs into Source (C3)
 - rAma
 - $x_1 = 1$
 - phala
 - $x_4 + x_6 = 1$
 - khaa
 - $x_7 = 1$
 - mohana
 - $x_3 + x_5 = 1$
 - KilOnA
 - $x_2 = 1$
 - xe
 - $x_8 = 1$

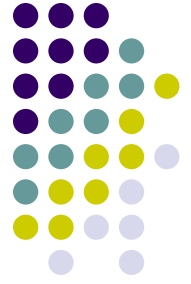
Solution Graph



- **rAma** **ne** **phala** **khaakara** **mohana** **ko** **KilOnA** **xiyA**
Ram 'ERG' fruit 'having eaten' Mohan 'DAT' toy gave
'Having eaten the fruit Ram gave the toy to Mohan'

Steps in Parsing

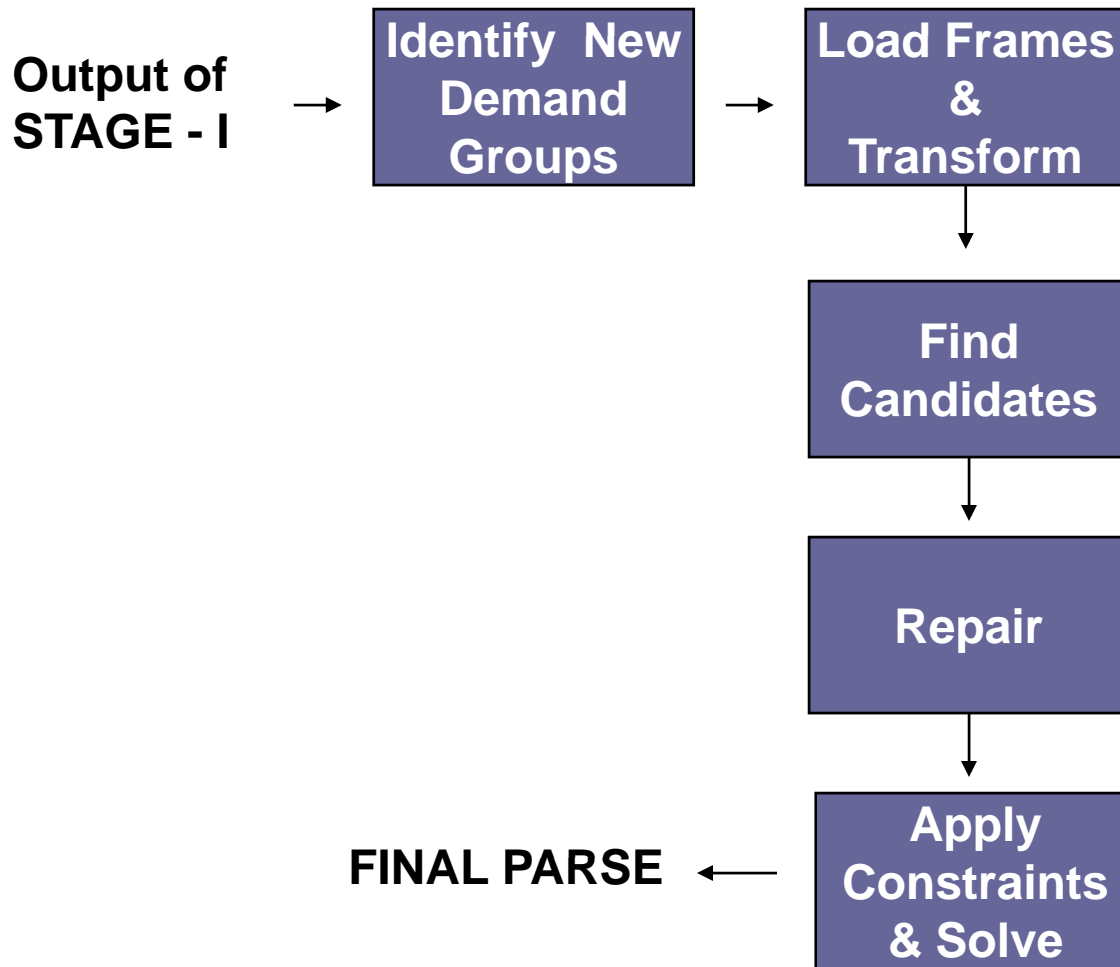
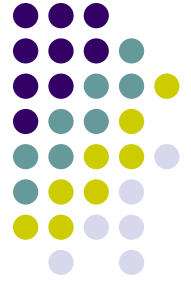




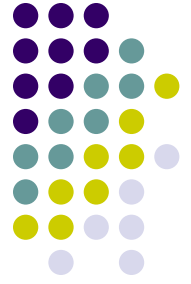
Stage - II

- Handles:
 - Conjuncts
 - Subordinating & Coordinating
 - Relative clauses
 - Complex predicates
- Basic constraints similar to Stage-I
 - New demand groups
 - New candidates

Steps (Stage II)



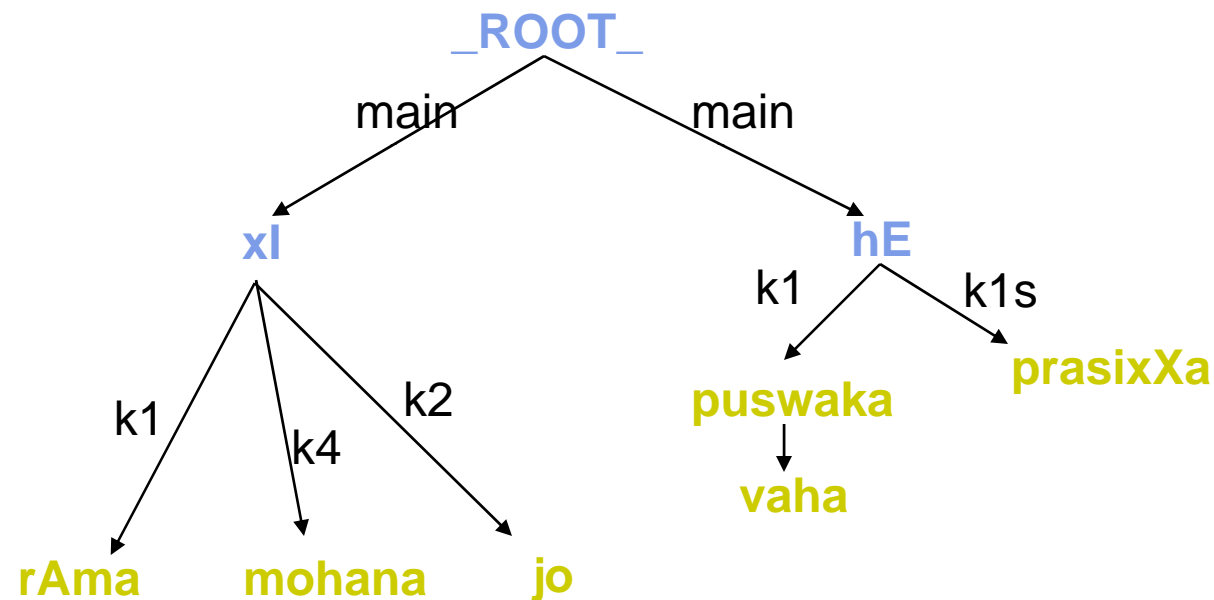
Example – Relative Clause



- vaha puswaka *jo rAma ne mohana ko xl hE* prasixXa hE
that book which Ram ERG. Mohana DAT. gave is famous is
'The book which Ram gave to Mohana is famous'

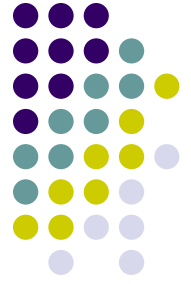


Output after Stage - I



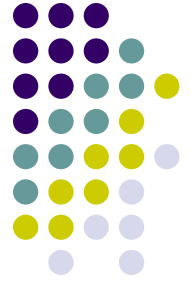
- vaha puswaka *jo rAma ne mohana ko xl hE* prasixXa hE
that book which Ram ERG. Mohana DAT. gave is famous is
'The book which Ram gave to Mohana is famous'

Identify the demand group



- xl 'give'
 - Main verb of the relative clause

Identify the demand group, Load and Transform DF



- *jo* ‘*which*’ transformation (special)
 - Transforms the demand frame of the main verb of the relative clause

arc-label	necessity	vibhakti	lextype	src-pos	arc-dir	oprt
nmod__relc	m	any	n	r l	p	insert



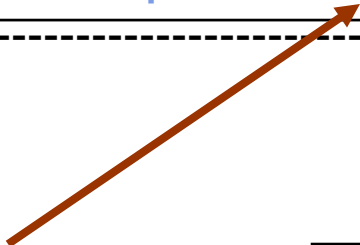
Karaka Frame

Main verb of relative clause

•vaha puswaka *jo* rAma ne mohana ko xl prasixXa hE |
that book which Ram ERG. Mohana DAT. gave famous is
'The book which Ram gave to Mohana is famous'

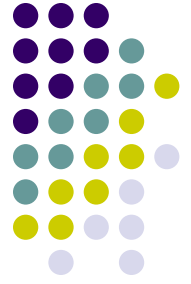
Transformed frame for *xe* after applying the *jo* transformation

arc-label	necessity	vibhakti	lextype	src-pos	arc-dir	oprt
nmod__relc	m	any	n	r	p	insert



New row inserted after transformation

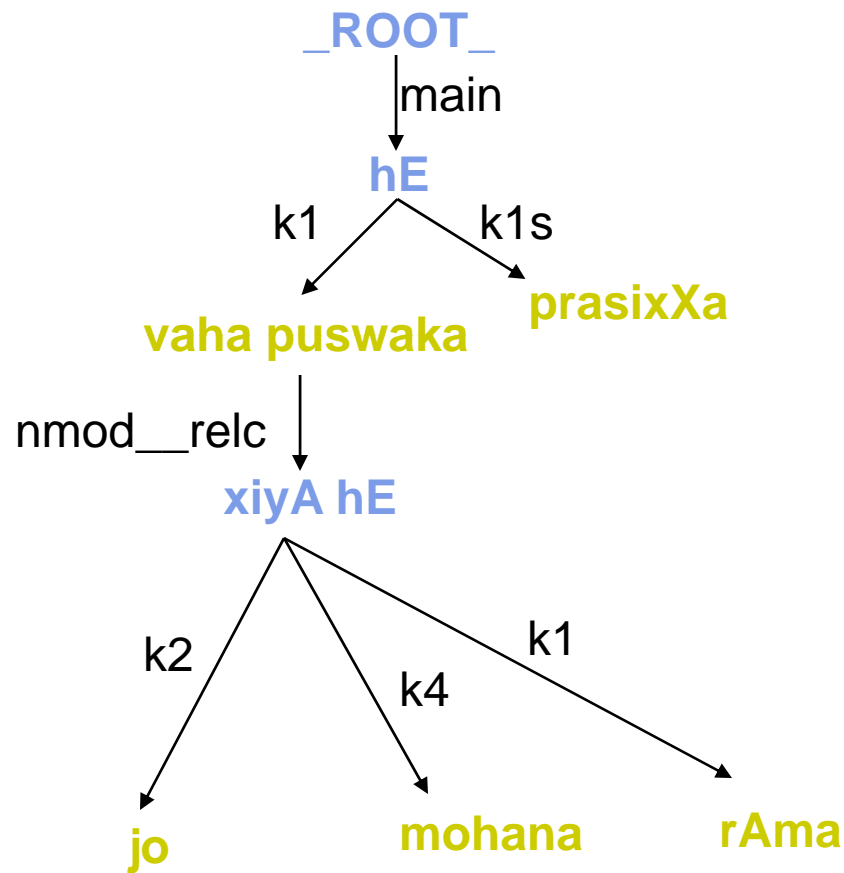
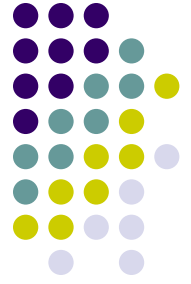
Possible candidates



nmod__relc

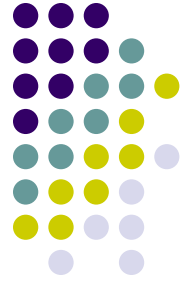
- vaha *puswaka* *jo rAma ne mohana ko* *xl hE* prasixXa hE |

Output after Stage - II

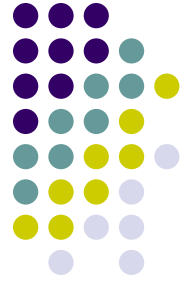


- vaha puswaka *jo rAma ne mohana ko xl hE* prasixXa hE
that book which Ram ERG. Mohana DAT. gave is famous is
'The book which Ram gave to Mohana is famous'

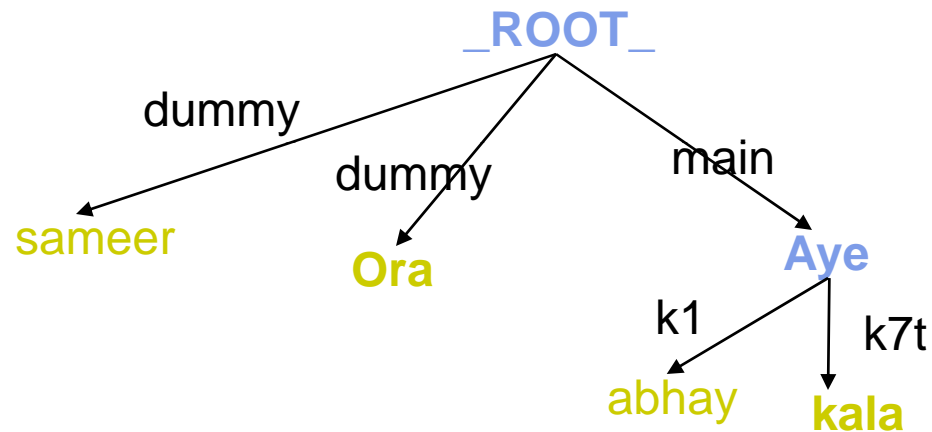
Example II – Coordination



- sameer Ora abhay kala Aye |
Sameer and Abhay yesterday came
'Sameer and Abhay came yesterday'

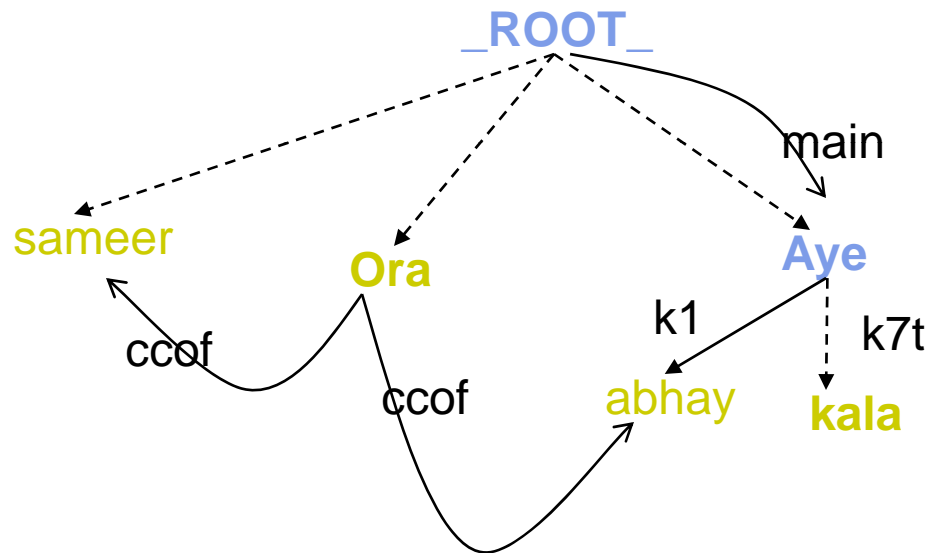
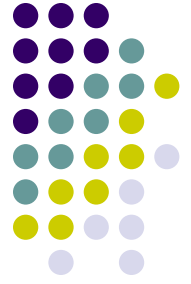


Output of Stage - I

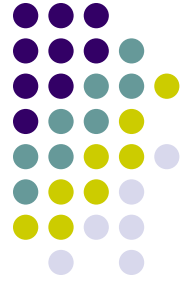


- sameer Ora abhay kala Aye |
Sameer and Abhay yesterday came
'Sameer and Abhay came yesterday'

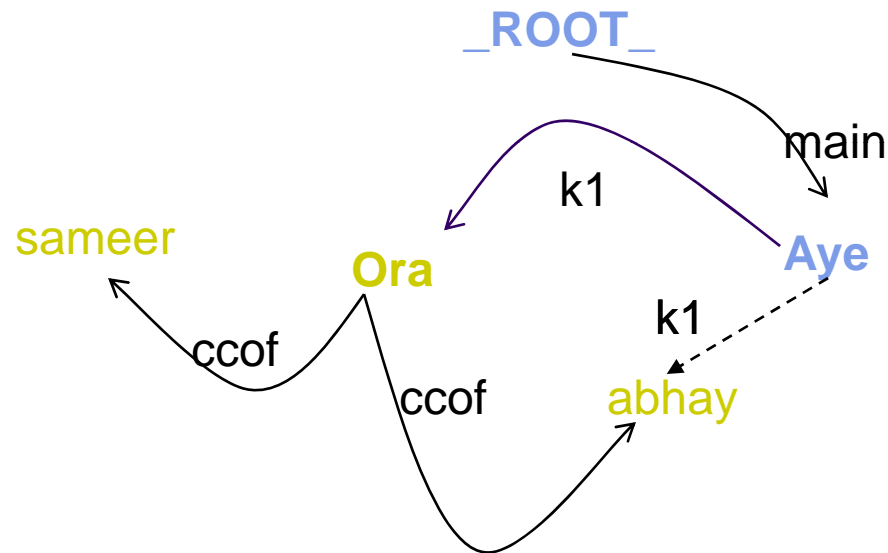
For Stage – II (Constraint Graph)



- sameer Ora abhay kala Aye |
Sameer and Abhay yesterday came
'Sameer and Abhay came yesterday'

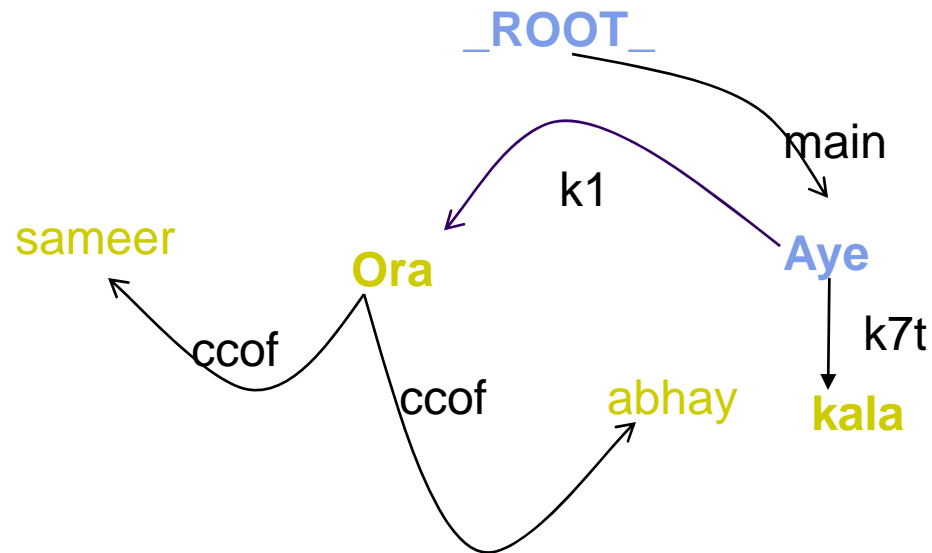


Candidate Arcs

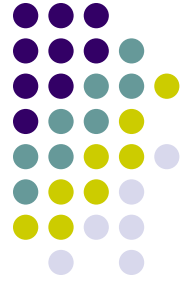


- sameer Ora abhay kala Aye |
Sameer and Abhay yesterday came
'Sameer and Abhay came yesterday'

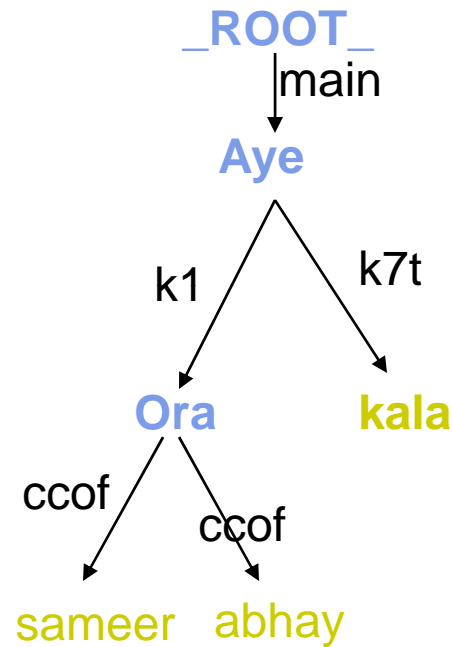
Solution Graph



- sameer Ora abhay kala Aye |
Sameer and Abhay yesterday came
'Sameer and Abhay came yesterday'



Parse tree



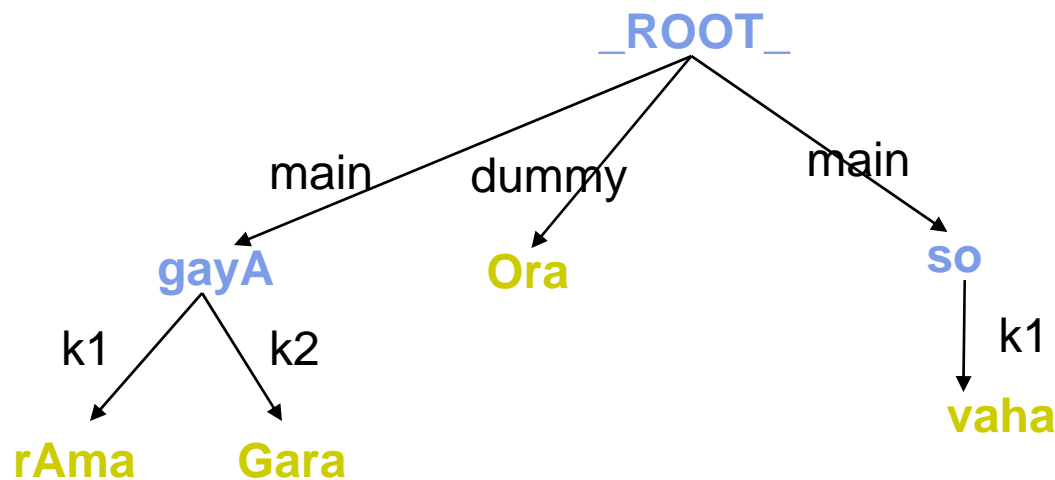
Output after Stage II

- sameer Ora abhay kala Aye |
Sameer and Abhay yesterday came
'Sameer and Abhay came yesterday'

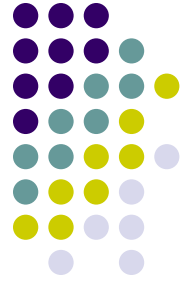


Finite Verb Coordination

- rAma Gara **gayA** **Ora** vaha **so** **gayA** |
Ram home went and he sleep went
'Ram went home and slept'



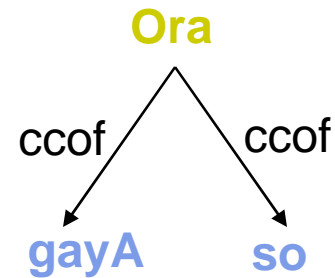
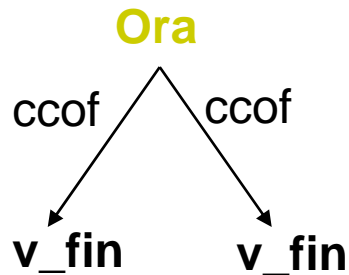
Output after Stage I



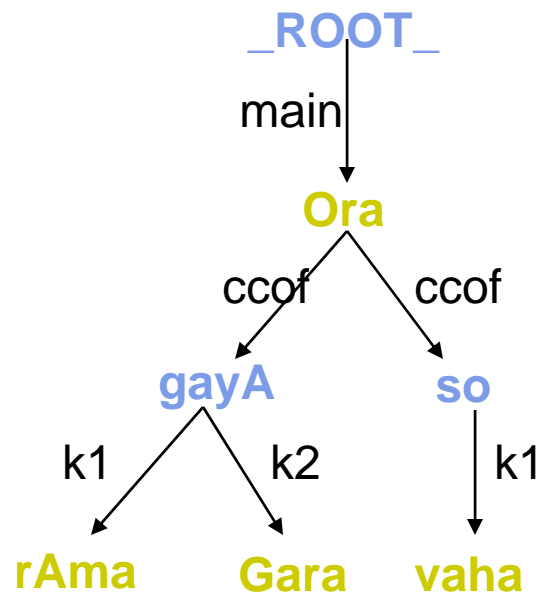
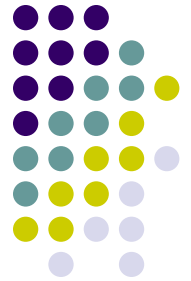
Karaka Frame - Ora

Finite

arc-label	necessity	vibhakti	lextype	src-pos	arc-dir	opr
ccof	tt	FINITE	v_fin	l	ds	-
ccof	tt	FINITE	v_fin	r	ds	-

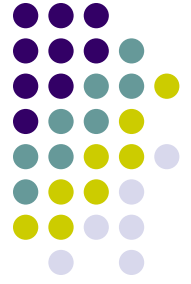


Finite Verb Coordination (Parse Tree)



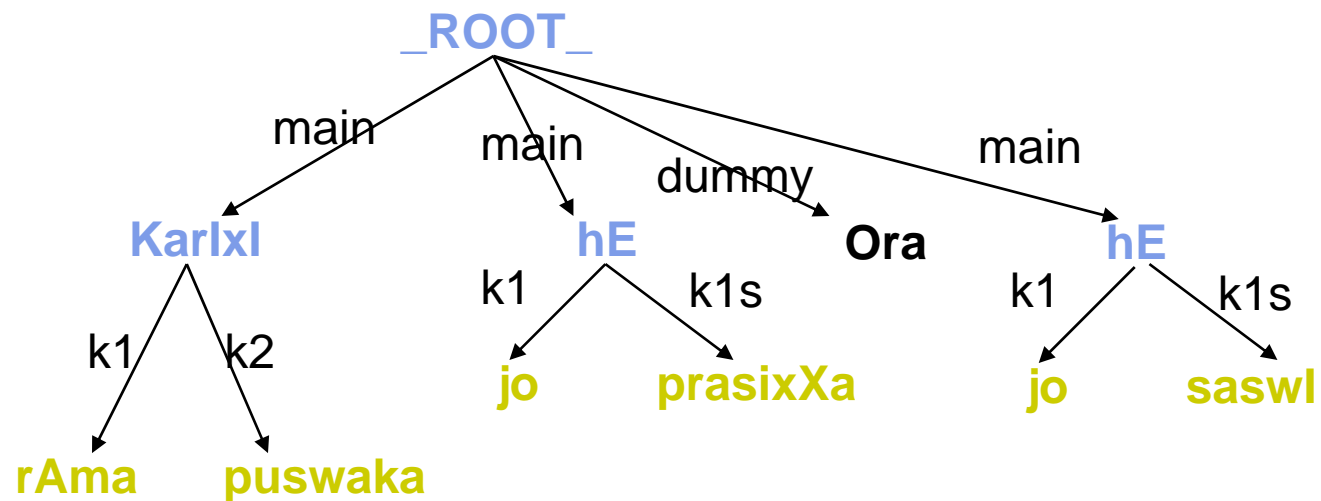
Output after Stage II

- rAma Gara gayA Ora vaha so gayA |
Ram home went and he sleep went
'Ram went home and slept'

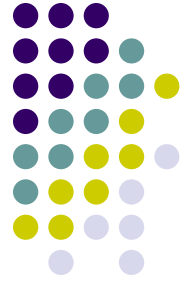


Relative Clause Coordination

- rAma ne vaha puswaka KarlxI *jo prasixXa hE* Ora *jo saswl hE*
‘Ram purchased the book which is famous and which is cheap’



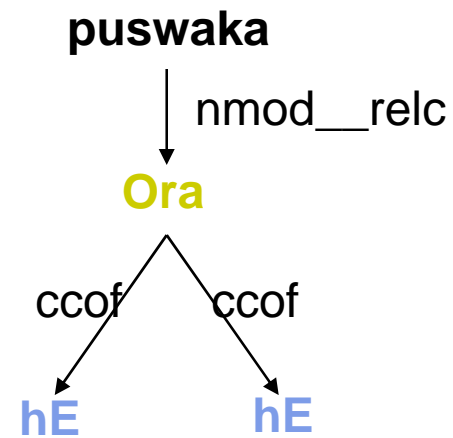
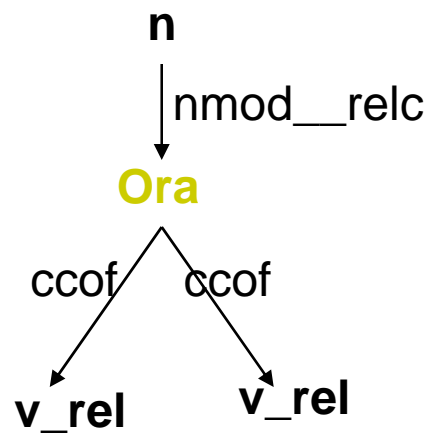
Output after Stage I



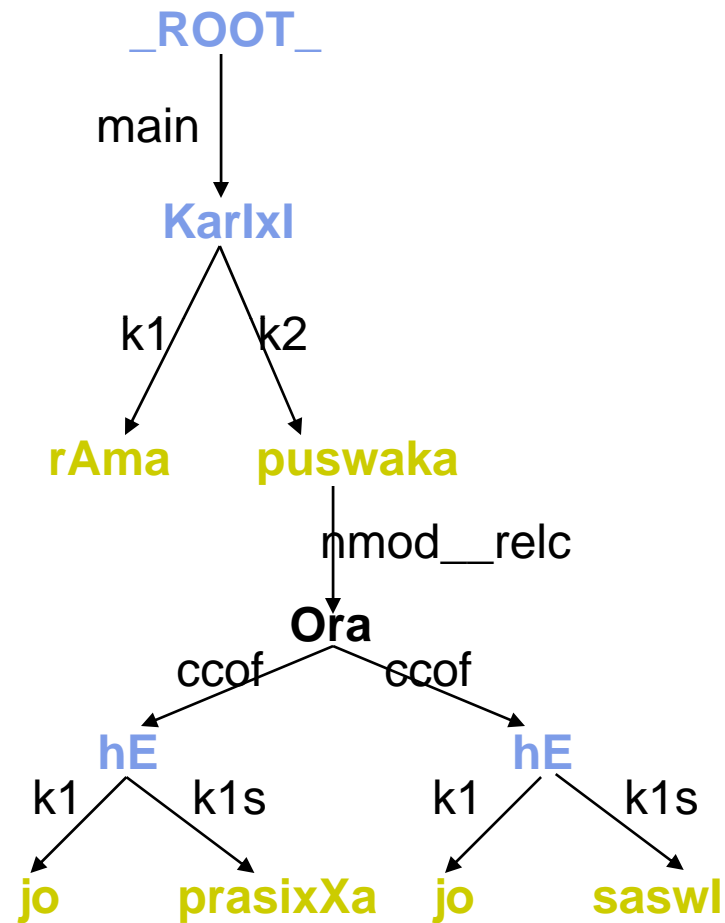
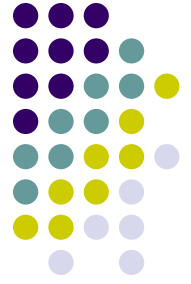
Karaka Frame - Ora

Relative Clause

arc-label	necessity	vibhakti	lextype	src-pos	arc-dir	oprt
ccof	m	FINITE	v_rel	l	ds	-
ccof	m	FINITE	v_rel	r	ds	-
nmod_relc	m	-	n	l r	sd	-



Relative Clause Coordination (Parse Tree)

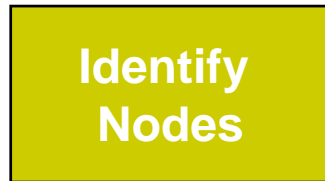


Output after Stage II

Steps (Stage II)

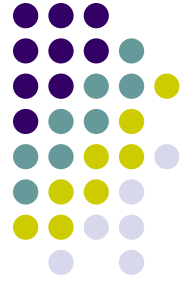


Output of
STAGE - I



FINAL PARSE ←

Constraint Graph Nodes (Stage II)



- Selected from the intermediate parse tree (Stage I)
- Set-I (demand nodes)
 1. Conjuncts
 2. Nearest verbal ancestor of 'jo' (usually just the parent)
 3. `_ROOT_`
 4. Children of `_ROOT_` other than (1) and (2).
 5. Other nodes which are added due to nodes in Set 2

Constraint Graph Nodes (Stage II)

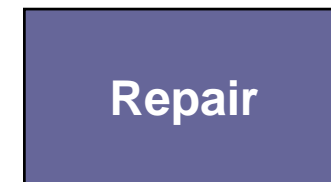
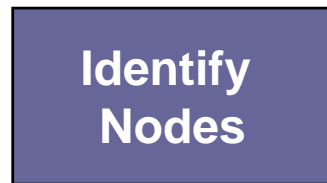


- Set-II (source nodes)
 1. Possible children and parents of conjuncts
 2. Possible heads of the relative clause.
- Identification of nodes in Set-II will generally trigger the repair.

Steps (Stage II)

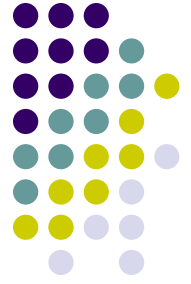


Output of
STAGE - I



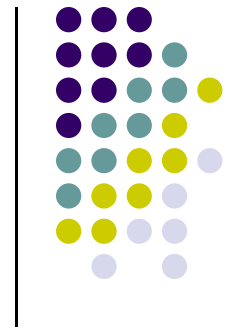
FINAL PARSE ←

Identify the demand group

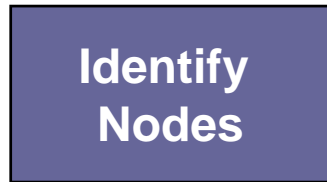


- Ora
- Aye

Steps (Stage II)



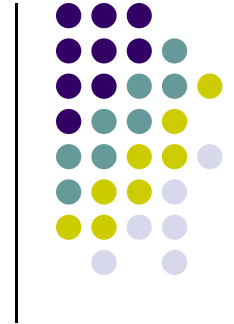
Output of
STAGE - I



FINAL PARSE ←



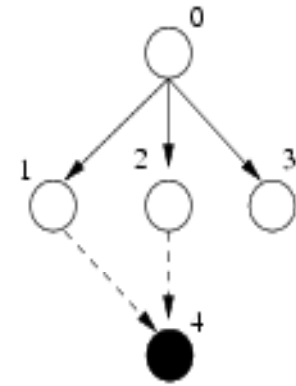
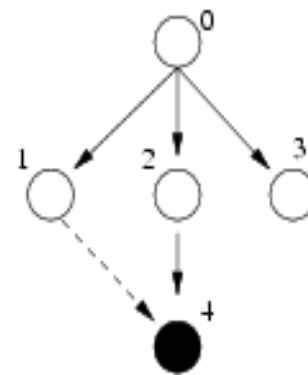
General Principles



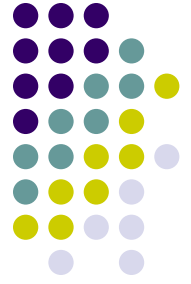
- Repair/Revision

1. Any node which becomes a potential child in stage 2, its arc to its existing parent is open to revision

- sameer Ora abhay kala Aye



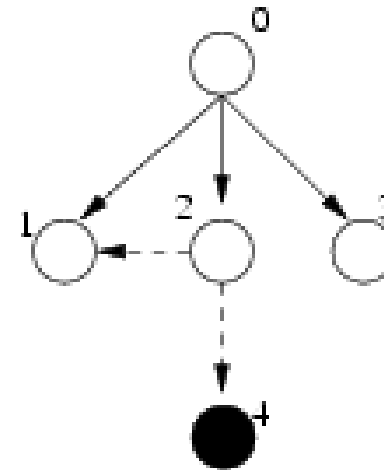
- Node 4 becomes potential child (of node 1)
- Its parent (node 2) is open to revision



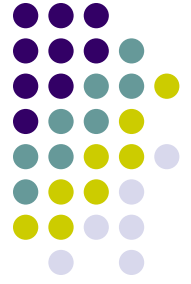
General Principles

- Repair/Revision after parse of stage I

2. Any node which becomes a potential parent must be re-looked at.

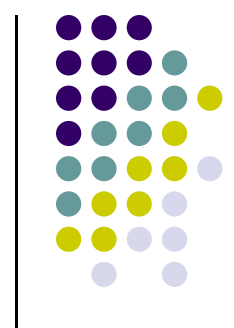


- sameer Ora abhay kala Aye
- Node 2 becomes potential parent (of 1)
- Its child (node 4) is open to revision



Algorithm

- Identify nodes of the constraint graph
 - From Set 1, and
 - From Set 2
- Remove all outgoing edges from `_ROOT_`.
- Find possible candidates for demand nodes present in Set 1 from Set 2
 - Parent candidate for finite verb
 - Parent and children for conjuncts
 - Children of `_ROOT_`
- Convert the formed constraint graph into integer programming (IP) problem.
- Solve the IP equations to get the possible solution parse.



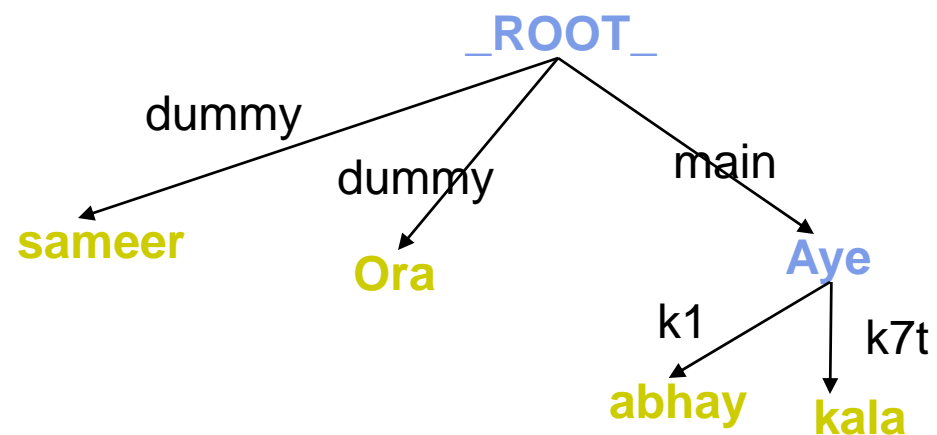
An example

sameer aura abhay kala aaye

'Sameer' 'and' 'Abhay' 'yesterday' 'came'

Sameer and Abhay came yesterday

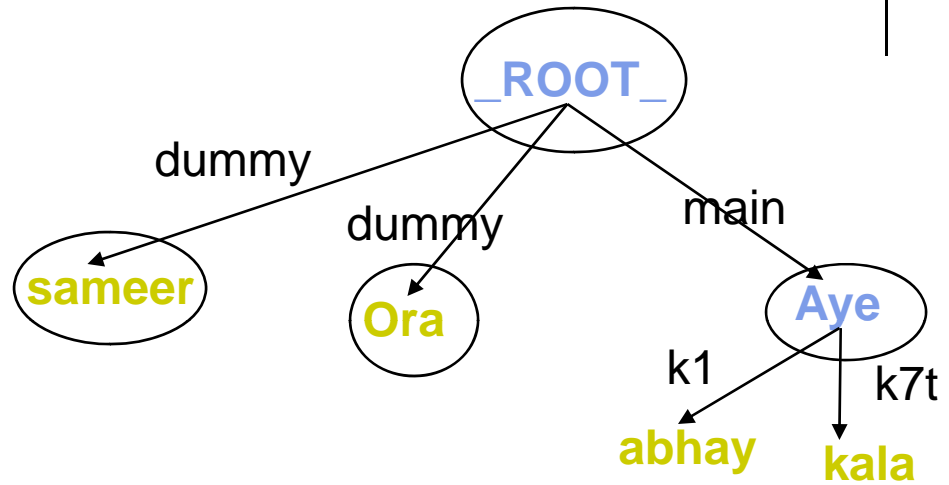
- Output after stage I



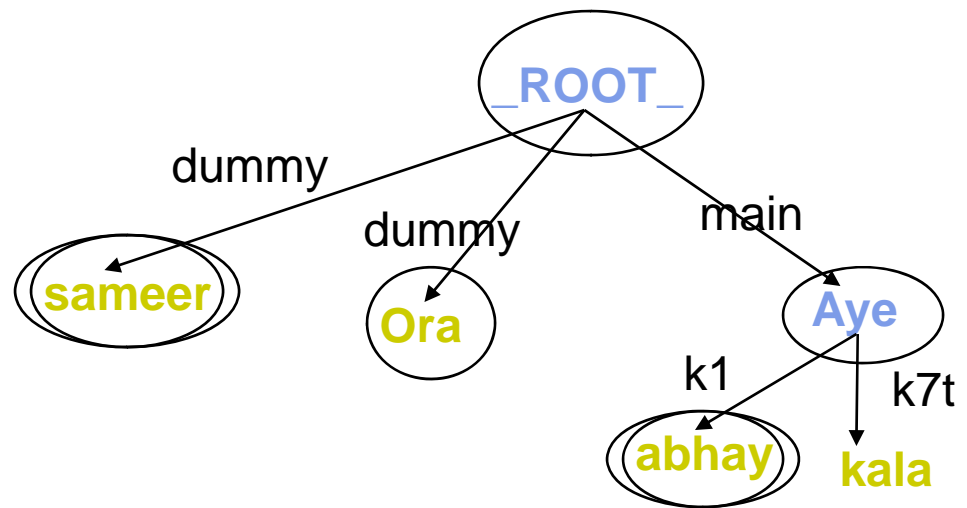
Identify Nodes

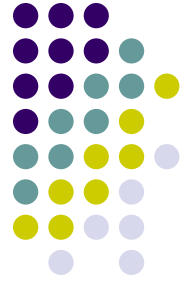


- Set 1 nodes



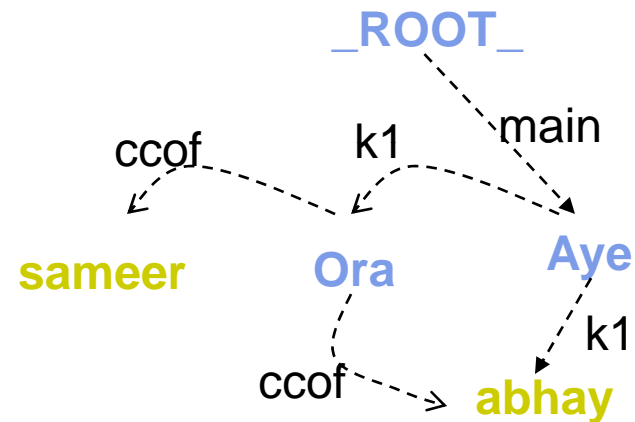
- Set 1 and Set 2



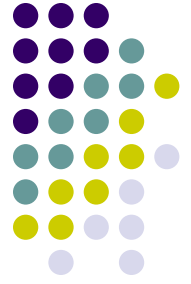


Constraint Graph

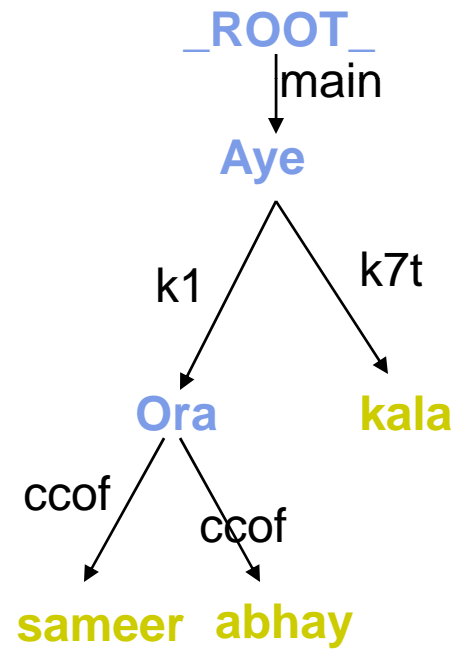
- New Constraint Graph
 - Ora, Aye and `_ROOT_` are the demand groups
- Note: '*kala*' remains attached to its parent '*aaye*' (does not show up in stage 2)

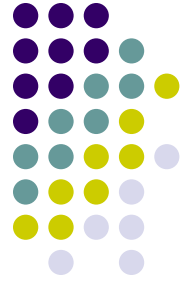


Example



- Final Parse





Types of complex sentences

- Relative clauses
 - Initial
 - Final
 - Medial
- Conjuncts
 - Coordination
 - Simple clause
 - Relative clause
 - Non-finite
 - Nominal, adjectival, adverbial
 - Subordination



Evaluation

- Two data driven parsers
 - Malt (version 0.4)
 - MST (version 0.4b)
- Tuned for Hindi
- Trained on a subset of a Hindi Treebank
- ~ 1800 sentences
 - average length of 19.85 words
 - 6585 unique tokens.
- Training set = 1185 sentences,
- Development = 268 sentences
- Test set = 220 sentences.



Overall performance

	UA	L	LA
CBP	86.1	65	63
CBP ^o	90.1	76.9	75
MST	87.8	72.3	70.4
Malt	86.6	70.6	68.0

Table 2.

UA: unlabeled attachments accuracy,

L : labeled accuracy

LA: labeled attachment accuracy

Core labels



		k1		k1s		k2		k3		k4	
		<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>
CBP''	P	74.9	74.4	71.5	71.5	54.0	53.7	66.6	66.6	28.5	28.5
	R	71.9	71.4	69.5	69.5	54.0	53.7	66.6	66.6	28.5	28.5
MST	P	77.2	75.2	54.5	54.5	55.9	52.3	29.6	29.6	0	0
	R	82.3	81.1	38.7	38.7	59.9	53.9	23.1	23.1	0	0
Malt	P	77.6	76.4	66.6	66.6	56.7	51.6	33.3	33.3	0	0
	R	81	80	43.4	43.4	58.2	52.9	16.6	16.6	0	0

Table 3(a) (P: Precision, R: Recall, L: Labeled accuracy, LA: Labeled attachment accuracy)

Core labels



		k7		r6		ccof		rele		nmod		vmod		main	
		<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>	<i>L</i>	<i>LA</i>
CBP''	P	75	71.1	85.5	84.2	98	98	66	66	41.6	33.3	81.1	77.7	91.3	91.3
	R	75	71.1	85.5	84.2	77	77	66	66	41.6	33.3	81.1	77.7	96.8	96.8
MST	P	69.7	64.3	86.6	85.9	85.3	80	90	80	25.0	22.5	82.5	79.7	98.9	98.9
	R	61.6	56.1	73.6	72.7	84.4	79.3	57.1	47.1	45.1	28.5	56.4	54.3	92	92
Malt	P	60.9	54.8	86.3	84.8	84.9	79.3	0	0	0	0	78.6	78.6	92.5	92.5
	R	53.1	47.8	74	72.7	83.5	78.1	0	0	0	0	53.9	53.9	92	92

Table 3(b) (P: Precision, R: Recall, L: Labeled accuracy, LA: Labeled attachment accuracy)



References

- R. Begum, S. Husain, A. Dhvaj, D. Sharma, L. Bai, and R. Sangal. 2008a. Dependency annotation scheme for Indian languages. *In Proceedings of IJCNLP-2008*.
- Akshar Bharati, Rajeev Sangal, T Papi Reddy. 2002. A Constraint Based Parser Using Integer Programming *In Proc. of ICON-2002*.
- J. Nivre, 2005. Dependency Grammar and Dependency Parsing. *MSI report 05133. Växjö University*.
- I. A. Mel'Cuk. 1988. *Dependency Syntax: Theory and Practice*, State University Press of New York.
- Tara Mohanan, 1994. *Arguments in Hindi*. CSLI Publications.
- S. M. Shieber. 1985. Evidence against the context-freeness of natural language. *In Linguistics and Philosophy*, p. 8, 334–343.
- R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. *In Proc. of HLT/EMNLP*, pp. 523–530.
- J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov and E Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2), 95-135.
- J. Nivre. 2006. *Inductive Dependency Parsing*. Springer.