Chapter 4

The Scheme

Following the grammar formalism described in chapter 3, a tagging scheme was developed to annotate dependency analysis for sentences in Indian languages, with a particular focus on Hindi.

4.1 Dependency Annotation Scheme

The Dependency Scheme [21] chosen for annotating Hindi Dependency Treebank (HyDT) is based on Paninian grammar framework. Dependency relations can refer to syntactic properties, semantic properties, or a combination of the two. Since in this framework, the meaning of a sentence unfolds by taking verb as the root node, verbs and their direct participants attain a central role. Panini’s grammar treats the direct participants of the action denoted by a verb root separately and calls them ‘karaka’. Following the framework, the scheme has two types of dependency relations:

A. karaka relations

B. Other dependency relations such as purpose, reason, adjectival, adverbial etc.

Apart from these dependency relations, the scheme also contains some arc labels which do not represent dependency relations. These are labels which mark certain arcs between two nodes which do not hold head-dependency relations (more details are given below).

The dependency tags are also classified as inter-chunk and intra-chunk. The inter-chunk dependencies are annotated manually whereas the intra-chunk dependencies are automatically marked, using a rule based tool, after the inter-chunk relations are marked. The current inter-chunk dependency tagset consists of about 40 tags. This number of tags was decided after taking into consideration different types of sentence constructions in Hindi. For all karaka relations, labels starting with a 'k' followed by a numerical have been used. The basic number of karakas are six but there are a number of relations which are either fine-grained types of karakas (such as k2p, k2g, etc.) or are in some way or the other related to a karaka (such as k1s, k2s, k1u, k2u, etc.) [28]. For dependency relations other than karakas, the labels start with an ‘r’. For non-dependency relations like ‘co-ordination’ and ‘complex predicates’, labels such as ‘ecof’ and ‘pof’ are used respectively. The ‘dependency relation type’ tree is given in Figure 4.1 below which shows the relations from coarser level to finer level on a modifier – modified paradigm [28]. We start with mod which is divided into vmod, nmod, jjmod, and rbmod. vmod is further sub-divided into varg and vad. varg contains karakas. vad, nmod, jjmod and rbmod contain non-karakas. We can keep sub-dividing coarser level relations into fine-grained relations if it is needed in future.
Figure 4.1 Dependency Relation Types

The classification shown in the above tree allows underspecification of particular relations in cases where a finer analysis is not very important for this level of annotation and is also more problematic for decision making for the annotators, i.e., in cases where things are not straight-forward about the marking of the relations, coarser level relations are marked instead of fine-grained relations, i.e., instead of marking karaka relations we mark vmod which is at the coarser level. Therefore, the labels such as k1, k2, etc., represent a fine-grained relations in the tree, whereas, labels such as vmod, nmod show an underspecified representation of the relations [28]. We have taken a simple example below, to show how inter-chunk dependency relations are marked within a sentence:

Ex- (4.1) raam ne mohan ko kitaab dii [28]

ram Erg. mohan Dat. book gave
‘Ram gave a book to Mohan.’

In the example 4.1 given above, raam ‘Ram’ is the ‘karta’ of the verb dii ‘gave’. mohan ‘Mohan’ is ‘sampradana’ (k4) and kitaab ‘book’ is the ‘karma’ (k2) of the verb dii. A dependency tree structure is formed by marking dependency relations in a sentence. The dependency tree structure for example 4.1 [28] is given below:
We mark modifier-modified (dependency) relations between the heads of the chunks. “A typical chunk consists of a content word surrounded by a constellation of function words, matching a fixed template. By contrast, the relationship between chunks is mediated more by lexical selection than by rigid templates” [6]. Chunks were introduced for the purpose of shallow parsing. A chunk is considered as a basic unit for marking the syntactico-semantic relations manually. Therefore, in our dependency tree [21]:

a. each node is a chunk, and

b. the edge represents the relations between the connected nodes labeled with the karaka or other relations.

For marking dependencies, we define chunk as a “minimal phrase with chunk internal dependencies not distorted”. “Internal dependencies are not distorted”, means that within a chunk, the elements would normally attach to the head. In cases, where a chunk member does not attach to the verb, but to another member, the information would be obtainable, i.e., from its POS (Part-of-Speech) tags. Thus, intra-chunk relations can be obtained with high degree of accuracy. Experiments have been conducted with high performance in automatically marking intra-chunk dependencies [29, 21]. The intra-chunk dependencies are marked automatically by using a rule based system. By and large the intra-chunk labels carry grammatical information such as: (i) postpositions attaching to the noun, and (ii) auxiliaries attaching to the verb. In Noun Phrases, there could be adjectives attaching to the noun. Relations such as karaka, purpose, reason etc., are normally marked between chunk heads. Hence the types of relations that are marked inter-chunk and the one’s that are marked intra-chunk are by and large different. Therefore, the number of dependency relations to be marked at the inter-chunk level is only 40. Thus, in manual annotation, the decision of annotating only inter-chunk dependency manually, helped the annotators in two ways:

(i) The number of dependencies to be marked per sentence is less. Moreover, since the number of labels is also fewer, the annotation decisions within a sentence becomes easier.

(ii) This also saves time.

Therefore, 4.2 below shows an example after it is manually annotated at the inter-chunk level. The tree structure of this example is shown in figure 4.3.
Ex- (4.2) raam ne (k1) mohan ko (k4) niili kitaab (k2) dii

‘Ram gave a blue book to Mohan.’

Figure 4.3 Inter-Chunk Level Dependency Tree structure

In the example 4.2 given above, raam ‘Ram’, mohan ‘Mohan’, kitaab ‘book’, and dii ‘gave’ are chunk heads whereas niili ‘blue’ is an adjective which occurs within the NP (Noun Phrase) chunk niili kitaab ‘blue book’ and therefore is left unannotated at this stage. Figure 4.4 given below, represents example 4.2 after the intra-chunk relations are automatically marked:

Figure 4.4 Intra-Chunk Level Dependency Tree structure

In the above figure, the relation between niili ‘blue’ and kitaab ‘book’ is of intra-chunk dependency relation.
4.1.1 karaka Relations

As mentioned earlier, karaka relations are participants directly involved in the action denoted by the verb. Paninian grammar talks about six karaka relations that participate in the action specified by a particular verb (discussed in Chapter 3) [36]. In deciding the karaka relations of the elements in a sentence, the semantics of the verb plays a major part and syntax also helps at the same time [28]. We briefly discuss the karaka relations with some examples below:

4.1.1.1 karta ('doer/agent/subject') (k1) [28]

As mentioned earlier, karta is the 'most independent' of all the karakas (participants) and it is one which carries out the action. It is different from the 'agent' as it does not always have volitionality. It is the locus of the activity denoted by the verb root. The activity resides in or springs forth from the karta. It is marked as k1. Following these properties of karta, ram ‘Ram’ is karta in the examples given below:

Ex-(4.3) raam (k1) roTii khaa-taa hai (doer of the action, in action verbs)


‘Ram eats roti.’

\[\text{Figure 4.5 Dependency Tree for karta}\]

Ex-(4.4) raam ne (k1) roTii khaa-yii

ram Erg. roti ate

‘Ram ate roti.’

Ex-(4.5) raam ko (k1) roTii khaa-nii pa.Dii

ram Dat. roti had-to-eat

‘Ram had to eat the roti.’
Ex-(4.6) **raam** dvaara (k1) **roTii** khaa-yii gayii (passive)
ram by roti ate Pasv.
'Roti was eaten by Ram.'

Ex-(4.7) **raam** kaa (k1) maana-naa hai ki kal barsaat hogii (genitive)
ram of belief is that tomorrow rain will-happen
'Ram believes that it will rain tomorrow.'

Ex-(4.8) **raam** (k1) achChaa hai (person/thing whose state is mentioned in stative verbs)
ram good is
'Ram is good.'

Ex-(4.9) **mohan** ko **raam** (k1) dikhaa (subject of unaccusative verb)
mohan Dat. ram appeared
'Ram was visible to Mohan.'

Ex-(4.10) (**raam** kaa yaha maananaa) (k1) sahii hai (clausal karta)
ram of this belief true is
'This belief of Ram is true.'

As we notice that **ram** ‘Ram’ is **karta**, however, the vibhakti (Hindi postposition / case markers) following **ram** ‘Ram’ are not always the same. A closer analysis of the examples shows us that all the examples are syntactically different. The above example 4.3, is a simple transitive verb. **ram** ‘Ram’ is **karta** and is in nominative and the verb agrees with it. **karta** does not always occur in nominative. In the example 4.4, **ne vibhakti** marks agent and although the verb doesn’t agree with **ram** ‘Ram’, **ne vibhakti** is clear marker of **karta**. In passive construction (refer example 4.6), **karta** (ram ‘Ram’) takes dvaaraa ‘by’ vibhakti. The example 4.7, is a genitive construction where **karta** (ram ‘Ram’) takes kaa ‘of’ vibhakti. In copula sentences, also stative verbs, **karta** is the person/thing whose state is mentioned (refer example 4.8). Both the examples 4.5 and 4.9, have **ko vibhakti**. Dative case marker **ko**, does not mark **karta** to mohan ‘Mohan’ in example 4.9, whereas it marks **karta** to ram ‘Ram’ in example 4.5. Dative case marker **ko**, marks **karta** when the TAM is naa_pa.Daa/nii_pa.Dii and naa_chaahiye. In the example 4.5, the TAM is nii_pa.Dii so **ram** ‘Ram’ with **ko vibhakti**, gets **karta karaka**. In the example 4.9, **ram** ‘Ram’ is **karta** and not mohan ‘Mohan’ (discussed in detail in chapter 5). In the example 4.10, the clause ‘**raam kaa yaha maananaa**’ ‘This belief of Ram’, is the **karta**. It is called as clausal **karta**.

The other types of **karta** are: **prayojaka karta** – pk1 (causer), **prayojya karta** - jk1 (causee), and **madhyasta karta** - mk1 (mediator-causer) which are used in causative constructions (discussed in detail in chapter 5).
4.1.1.2  *karma* (‘object/patient’) (k2) [28]

As shown in section 3.1.2 (chapter 3), *ashraya* or the locus of the result denoted by the verb root is called *karma*. It is the object/patient of the verb. It is most desired to be attained by the *karta*. It is marked as *k2*. Following are the examples of *karma*:

**Ex-(4.11)**  
*raam ne [roti (k2)] khaayii (active)*  
ram Erg roti ate  
‘Ram ate roti.’

**Ex-(4.12)**  
*divaalii ke din khuub miThaaii (k2) khaayii gayii (passive)*  
diwali of day lots-of sweets ate Pasv.  
‘Lots of sweets were eaten on the day of Diwali’

**Ex-(4.13)**  
*raam ne bataayaa (ki vaha kal nahiiM aayegaa) (k2) (Vakya-karma-sentential object)*  
ram Erg. told that he tomorrow not come.Fut.  
‘Ram told that he will not come tomorrow.’

**Ex-(4.14)**  
*raam ne [ravi ko (k2)] maaraa*  
ram Erg ravi Acc. beat  
‘Ram hit Ravi.’

![Dependency Tree for karma](image)

*Figure 4.6* Dependency Tree for *karma*

Most often, *karma* takes either zero vibhakti or a ‘ko’ vibhakti (Hindi postpositions / case markers). When both *karta* and *karma* take a zero vibhakti in a sentence, then the noun which does not agree with the verb would be marked as *karma* (refer example 4.3). In the above given example 4.3, *roTii ‘roti’* is the *karma* as it doesn’t agree with the verb *khaataa hai ‘eats’*. If a *karta* is followed by a vibhakti and there is another noun in the sentence which agrees with the verb then that noun becomes the *karma*. In the above given example 4.11, the *karta* (*raam*) is followed by a vibhakti (*ne*) so *roTii ‘roti’* becomes
the karma as it agrees with the verb in terms of person, number, and gender. karma also takes 'ko' vibhakti when it is a human noun or when it indicates definiteness. In the example 4.14, ravi ‘Ravi’ is karma as it is a human noun. In passive constructions, the noun which agrees with the verb becomes the karma. In the example 4.12 which is a passive construction, miThaaii ‘sweets’ is the karma as it agrees with the verb. In the example 4.13, ‘ki vaha kal nahiM aayegaa’ ‘that he will not come tomorrow’ is the vakya-karma, i.e., sentential object. The subordinate clause ‘ki vaha kal nahiM aayegaa’ ‘that he will not come tomorrow’ attaches to the verb bataayaa ‘told’ of the main clause and the relation marked between them is marked as k2.

Hindi has complex, but robust agreement rules. karta and karma agree with the verb. karta takes zero vibhakti (Hindi postposition / case markers) when it agrees with the verb and it takes the following vibhaktis when it doesn’t agree with the verb: ne, ko, se, dvaaraa. Therefore, a mapping between vibhakti and TAM (tense, aspect and modality) is helpful in identifying karta and karma [28]. Sometimes Hindi also shows agreement between noun and verb within a conjunct verb.

Ex- (4.15) raam ko siita kaai (k2) yaad aayii
          ram Dat sita Gen. remembrance came
          'Ram remembered Sita.'

In the above example 4.15, there is an agreement between the noun yaad ‘remembrance’ and verb aayii ‘came’ within a conjunct verb.

4.1.1.3 karana (‘instrument’) (k3) [28]

karana karaka is an instrument used in the action expressed by a verb root. The vyapara or activity of the karana helps in achieving the phala or result immediately. The above statement means that karana helps in accomplishing the action denoted by the verb root. The noun which denotes an instrument need not be a physical object. By default karana karaka takes se vibhakti (Hindi postpositions / case markers). It is marked as k3. In the given below examples 4.16 and 4.17, chaakuu ‘knife’ and paanii ‘water’ are marked as karana karaka respectively.

Ex-(4.16) raam ne chaakuu se (k3) aam kaaTaa
            ram Erg, knife with mango cut
            'Ram cut the mango with a knife.'
In example 4.16 given above, *chaakuu ‘knife’* is *karana* because it is an instrument with which the result of the main action *kaaTaa ‘to cut’* is achieved. Here the main action is ‘cutting’ and the result is `pieces of the mango’. In example 4.17, the noun *paanii ‘water’* is *karana* and it is not a physical object. Here *paanii ‘water’* is instrumental in achieving the action.

4.1.1.4 *sampradana* (*recipient*) (*k4*) [28]

It is the receiver or beneficiary of an action. *sampradana* is the person/object for whom the *karma* is intended. It is the one where the *karma* reaches. It is not always goal. It occurs only when there is some transaction and the transaction could be either active or passive. It is marked as *k4*. We also have *k4a* which is the subtype of *k4*. It means *anubhava karta* which means experiencer of the action. It doesn’t actively participate in the action.

Ex-(4.18) *raam ne mohan ko* (*k4*) *miTaaii dii*

ram Erg. mohan Dat. sweets gave

‘Ram gave sweets to Mohan.’

---

**Figure 4.7** Dependency Tree for *karana*

Ex-(4.17) *raadha ne paanii se* (*k3*) *botala ko bharaa*

raadha Erg. water with bottle Acc. filled

‘Radha filled the bottle with water.’

---

In example 4.16 given above, *chaakuu ‘knife’* is *karana* because it is an instrument with which the result of the main action *kaaTaa ‘to cut’* is achieved. Here the main action is ‘cutting’ and the result is `pieces of the mango’. In example 4.17, the noun *paanii ‘water’* is *karana* and it is not a physical object. Here *paanii ‘water’* is instrumental in achieving the action.

4.1.1.4 *sampradana* (*recipient*) (*k4*) [28]

It is the receiver or beneficiary of an action. *sampradana* is the person/object for whom the *karma* is intended. It is the one where the *karma* reaches. It is not always goal. It occurs only when there is some transaction and the transaction could be either active or passive. It is marked as *k4*. We also have *k4a* which is the subtype of *k4*. It means *anubhava karta* which means experiencer of the action. It doesn’t actively participate in the action.

Ex-(4.18) *raam ne mohan ko* (*k4*) *miTaaii dii*

ram Erg. mohan Dat. sweets gave

‘Ram gave sweets to Mohan.’
Figure 4.8 Dependency Tree for sampradana

Ex-(4.19) raam ne mohan se (k4) yaha bolaa

ram Erg. mohan to this told
‘Ram told this to Mohan.’

Ex-(4.20) raam ne mohan ko (k4) yaha baat bataayii

ram Erg. mohan to this matter said
‘Ram said this (matter) to Mohan.’

sampradana karaka takes ko vibhakti (Hindi postposition / case markers) in Hindi. In all the above given examples (4.18 - 4.20), mohan ‘Mohan’ is the sampradana as it is the receiver of the action. sampradana is also used in communication verbs. In communication verbs, sampradana is the person to whom the communication reaches or the communication is intended. In communication verbs, mohan ‘Mohan’ which is the sampradana is taking se and ko vibhakti in the examples 4.19 and 4.20 respectively. sampradana takes se vibhakti for some communication verbs and ko vibhakti for the other communication verbs.

4.1.1.5 apadana (‘source’) (k5) [28]

It indicates source of an activity. apadana karaka is the one which remains fixed in the action involving separation. It is the point of departure, i.e., there will be a movement away from a point. The participant which remains stationary when the separation takes place is marked as apadana. Source could be concrete or abstract. It is marked as k5.
Ex-(4.21) raam ne chammach se kaTorii se (k5) aisa-kriim khaayii
ram Erg spoon with bowl from ice-cream ate
‘Ram ate ice-cream from the bowl with a spoon.’

Figure 4.9 Dependency Tree for apadana

Ex-(4.22) chor pulisa se (k5) bhaagataa hai
thief police from run.Impf. be.Pres.
‘The thief runs away from the police.’

apadana karaka also takes a se vibhakti (Hindi postposition / case markers) in Hindi. Since ‘se’
postposition in Hindi is overloaded, we need to search for additional cues in cases where se occurs as a
vibhakti. We can take verb semantics as a cue for apadana karaka. So if the verb denotes some motion,
then the noun which indicates point of departure would be marked with ‘se’ and that would be apadana
karaka. In the above examples 4.21 and 4.22, kaTorii ‘bowl’ and pulisa ‘police’ are marked as
apaadaan. They both have se vibhakti.

I. Emotional verbs take apaadaan karaka.

The entity which triggers certain emotions such as anger and happiness are annotated as k5. In
Hindi, the emotional verbs such as gussaa honaa `to be angry', Khusha honaa `to be happy' take apadana
karaka. In the example 4.23 given below, harii ‘Hari’ is apaadaan as he denotes the source of anger.
harii ‘Hari’ would be the point of departure for the emotion of anger triggered in raam ‘Ram’ and thus,
harii ‘Hari’ will be marked as ‘k5’ [28].

Ex-(4.23) raam harii se (k5) gussaa hai
ram hari from anger is
‘Ram is angry with Hari’
II. The entity from which the information has to be obtained is marked as k5.

usase ‘him-Abl’ is apaadaan in example 4.24 as raam ‘Ram’ is the entity from which information is being obtained.

Ex-(4.24) maine raam se (k5) eka prashna puuChaa
I-Erg ram Abl one question asked
‘I asked a question to Ram.’

4.1.1.6 adhikaraṇa (k7) [28]

adhikaraṇa karaka is the locus of karta or karma. It is the location in time/place/other. It supports the karta or karma in space or time and other than space or time. The classification of adhikaraṇa karaka is captured in following ways:

(i) kaalaadhikaraṇa ('location in time') (k7t)
(ii) deshadhikaraṇa ('location in space') (k7p)
(iii) vishayadhikaraṇa ('location elsewhere') (k7)

4.1.1.6.1 kaalaadhikaraṇa ('location in time') (k7t) [28]

The participant indicating the time of action is marked as ‘k7t’.

Ex-(4.25) kal (k7t) raam aayaa thaa
yesterday ram came be.Pst.
‘Ram had come yesterday.’
It is not always that kaalaadhikarana takes a vibhakti (Hindi postposition / case markers). In example 4.27, zamaane ‘period’ is denoting time so it would be marked as kaalaadhikarana. Here, it is taking meM ‘in’ vibhakti. Time expressions such as pahale ‘before/first’, kala ‘yesterday’, when occur as kaalaadhikarana, do not take any vibhakti. In example 4.25, kal ‘yesterday’ is kaalaadhikarana and has no vibhakti. In some cases it takes ko and para ‘on’ vibhaktis. In example 4.26, shaam ‘evening’ or samaya ‘time’ denoting time of action would be marked as kaalaadhikarana. Here shaam ‘evening’ is taking ko vibhakti and samaya ‘time’ is taking par vibhakti.

4.1.1.6.2 deshadhikarana ('location in space') (k7p) [28]

The participant which indicates the location of karta or karma at the time of action is marked as k7p. It refers to a physical place and not an abstract place.

Ex- (4.28) meza para / thailii meM (k7p) kitaab hai
table on bag in book is
‘The book is on the table / The book is in the bag.’
Ex- (4.29) **raam vahaa.N (k7p) kha.Daa hai**

ram there standing is

‘Ram is standing there.’

deshadhikarana sometimes takes a vibhakti and sometimes it doesn’t. It takes meM ‘in’ or para ‘on’ vibhakti. In example 4.28, meza para ‘on the table’ or thailii meM ‘in the bag’ indicating place would be marked as deshadhikaran. meza ‘table’ is taking par vibhakti and thailii is taking meM vibhakti in the example 4.28. In the example 4.29, vahaa.N ‘there’ which is indicating place of action is marked as deshadhikara. Here vahaa.N ‘there’ doesn’t take any vibhakti.

**4.1.1.6.3 viShayaadhikarana (‘location elsewhere’) (k7) [28]**

It is considered as the location other than time and place. Location or place which is not concrete is marked as viShayaadhikarana. It is also used to denote ‘location in topic’ as well as ‘events’. It is marked as k7.

Ex-(4.30) **usane svatantrataa samgraam meM (k7) bhaag liyaa**

he-Erg independence movement in part took

‘He took part in the independence movement.’

Ex-(4.31) **raam ke man meM (k7) kaDvaahaT hai**

ram Gen. mind in bitterness is

‘Ram is full of hatred.’

Ex-(4.32) **raam kaa man hindustaan meM (k7) hai**

ram Gen. mind India in is

‘Ram is mentally in India.’

Ex-(4.33) **raam filmii viSayoM par (k7) charchaa kar rahaa thaa**

ram film topic on discussion do Prog. was

‘Ram was discussing films.’

It takes meM and par vibhaktis. In example 4.30, samgraam ‘movement’ denoting an event is marked as viShayaadhikarana. In the example 4.31, man ‘mind’ is marked as k7 as it denotes an abstract location. In the example 4.32, hindustaan ‘India’ is marked as k7. Here it is not marked as k7p though it is an actual physical place, because the entity man ‘mind’ which is in hindustaan ‘India’ is not a physical entity. viShayaadhikarana in the examples 4.30-4.32, take meM vibhakti. In the example 4.33, filmii viSayoM ‘film matters’ is marked as k7 as it denotes location in topic and takes par vibhakti.
4.1.2 Other Dependency Relations

These are relations other than karakas. Apart from 'karaka' relations, a sentence contains other types of relations as well. For example, relations such as, purpose, reason, genitive, associatives, comparatives, adjectives, etc. These relations are also marked within this scheme. Some of them are given below:

- r6 – Genitive
- rt – Purpose
- rh – Reason
- rad – Address

4.1.2.1 shashThi ('genitive/possessive') (r6) [28]

It is the genitive/possessive relation between two nouns. It is marked as r6. It mostly takes kaa ‘of’ vibhakti (Hindi postpositions / case markers) and it agrees with the noun it modifies in number and gender. In example 4.34, raam ‘Ram’ is r6 and takes kaa ‘of’ vibhakti. Here, the vibhakti kaa ‘of’ agrees with the noun kitaab ‘book’ which it modifies. The vibhakti kaa ‘of’ has feminine gender and singular number since kitaab ‘book’ has these features.

Ex-(4.34) raam kii (r6) kitaab
    ram of book

4.1.2.2 tadarthya ('purpose') (rt) [28]

It is the purpose of an action. It is marked as rt. It takes ke_liye ‘for’ vibhakti. In the examples 4.35 and 4.36, mohan ‘Mohan’ and jaanaa ‘going’ denotes purpose respectively. So they are marked as tadarthya and they both take ke_liye ‘for’ vibhakti.

Ex-(4.35) raam ne mohan ke liye (rt) miThaaii kharidii
    ram Erg mohan for sweets bought
    ‘Ram bought sweets for Mohan.’
4.1.2.3 **hetu ('reason') (rh)** [28]

It is the reason/cause of an activity. It is marked as *rh*. It takes *ke_kaaraNa*, ‘because of’, *kii_vajaha_se* ‘because of’, and *se* ‘due to’ vibhaktis. In the example 4.37, *mohan* ’Mohan’ is *hetu* as it denotes the reason of the action and takes the complex *vibhakti kii_vajaha_se / ke_kaaraNa* ‘because of’.

Ex-(4.37) *raam mohan kii vajaha se / ke kaaraNa (rh) leT ho gayaa*

‘Ram became late because of Mohan.’

4.1.2.4 **rad (address terms)** [28]

Terms such as *shriimaanjii, paMDitajii* etc. are considered as the address terms. In the example 4.38 given below, *maa.N* ’mother’ is marked as *rad* as it is an address term.

Ex-(4.38) *maa.N, (rad) mujhe khaanaa chaahiye*

‘Mother, I want food.’

As mentioned earlier, there are certain relations which are not exactly dependency relations but are required for representing the sentence structures. We call them as non-dependency relations. Below, we discuss the non-dependency relations used in the scheme.

### 4.2 Non-dependency relations

These relations which do not fall under 'dependency relations' directly, but are required for showing the dependencies indirectly. These labels mark certain arcs between two nodes which do not have head dependency relations. They are used for representing labels such as, ‘ccofo’, ‘pof’, and ‘fragof’ which are present in the tagging scheme to represent *co-ordination, complex predicates, and discontiguous elements (Fragment of)* respectively.
4.2.1  

**ccof (Conjunctions)**

It is used for coordinating and subordinating conjunctions. It is marked as `ccof`. ‘`ccof`’ is not exactly a dependency relation. It is a non-dependency relation. Conjunctions are classified as function words and they would normally not become a node in the tree. However, they cannot be left out of the tree; they need to be represented in the tree. However, since conjunctions do not reflect dependency, the labels for the arcs coming out from conjuncts do not have a dependency label. Thus, the label ‘`ccof`’ is classified in our scheme under the non-dependency labels category. This tag is used for annotating coordinating as well as subordinating conjunctions. A chunk having `ccof` tag should be attached to a conjunct. A conjunct chunk should have children of the same type i.e. same chunks (between NP, VG chunks etc.) [28]

**Example (4.39)**

```plaintext
raam ne  khaaanaa  khaayaa  aur (ccof)  siita ne paanii  piyaa  (co-ordinating conjunction)
```

Ram Erg. food ate and sita Erg. water drank

‘Ram ate food and Sita drank water.’

**Example (4.40)**

```plaintext
raam ne  kahaa  ki (ccof)  vo  kal  aayegaa  (sub-ordinate Conjunction)
```

Ram Erg. said that he tomorrow will-come

‘Ram said that he will come tomorrow.’

In the example 4.39, `aur` ‘and’ is co-ordinating conjunction and in the example 4.40, `ki` ‘that’ is subordinating conjunction (more details on this section in chapter 5). Coordinating conjunction `aur` ‘and’ is connecting the two verbs `khaayaa` ‘eat’ and `piyaa` ‘drink’ by the relation `ccof` in the example 4.39. In the example 4.40, the relation between the subordinating conjunct `ki` ‘that’ and the verb of the subordinating clause `aayegaa` ‘will come’ is `ccof`.

4.2.2  

**pof (Conjunct verbs)**

**Conjunct verbs** in Hindi are formed by combining a noun or an adjective with a verb. A conjunct verb behaves as a single semantic unit. The noun/adjective + verb sequence of the conjunct verb is placed in one chunk. There occurs a problem in cases where the noun of a conjunct verb takes its own modifiers. For this purpose, the conjunct verb is broken into two chunks, Noun/Adjective chunk and Verb chunk. To capture the relation between noun and the verb of the conjunct verb, we are using a special tag ‘`pof`’ to
mark the conjunct verbs. The dependency relation of prashna with kiyaa will be POF (‘Part OF’ relation), i.e. the noun or an adjective in the conjunct verb will have a POF relation with the verb.

Ex- (4.41) maine usase eka prashna (pof) kiyaa
I-Erg him-Abl one question did
‘I asked him a question.’

In the example 4.41, prashna kiyaa ‘questioned’ is conjunct verb which is made of the noun prashna ‘question’ and verb kiyaa ‘did’. The relation marked between the noun prashna ‘question’ and kiyaa ‘did’ of the conjunct verb is pof (more details on this section in chapter 5 and chapter 7).

4.2.3 fragof (Fragment of) [28]

There are cases where an element is inserted between the parts of a single chunk or sometimes when the main part of the chunk is dropped. ‘fragof’ (fragment of) is used to mark the relation between the two members of the same chunk.

Ex- (4.42) bhaakapaa (maaovaadii) ke (fragof) raamabachana yaadava ko girafataar kara liyaa
BKP (maoist) of rambacana yadav Acc. arrest do reflx-perf
gayaa go-perf
‘Apart from this, Rambacana Yadav of BKP (Maoist) was arrested.’ [28]

In the above example, (maaovaadii) ‘maoists’ has been inserted between the noun bhaakapaa ‘BKP’ and postposition ke ‘of’ which is a single chunk. Because of this, these two elements are chunked separately, i.e., bhaakapaa ‘BKP’ as noun chunk and ke ‘of’ as FRAGP (Fragment phrase) chunk. The relation between ‘ke’ ‘of’ and ‘bhaakapaa’ ‘BKP’ will be marked as fragof [28].

Thus, a fairly detailed scheme based on Computational Paninian Grammar (CPG) framework for annotating Hindi sentences for dependency relations has been developed. This scheme is being currently used for developing the Hindi dependency Treebank (HyDT) and also for the multi-layered/multi-representational Hindi and Urdu Treebank [42, 188]. This scheme has also formed the basis of developing guidelines/scheme for the Urdu dependency Treebank which is also in the making at IIIT-H (IIIT-Hyderabad).

The scheme has also led to the exploration of dependency annotation for other Indian languages (Telugu [182], Bangla) and English [57].
4.3 Annotation Procedure

4.3.1 Syntactic Annotation

Although the actual task was dependency analysis of the sentences, however, a pre-requisite of dependency annotation was POS tagged, chunked, and morphologically analysed corpus. So, this dependency annotation involved, is carried out over and above POS tagged, chunked, and morphologically analysed corpus. Therefore, it was decided to annotate the following levels of linguistic information in the Hindi Treebank [28]:

a) Morphological information

b) POS tagging (Part-of-Speech-lexical level)

c) Chunks: Identification of minimal constituents and their heads (phrasal level)

d) karaka or dependency relationships: Marking relations across chunks, i.e., head to head relations (sentential level–syntactico-semantic)

e) Sentence type

f) Voice type

g) Coreference in specific cases

SSF format was chosen to represent the annotated data. The Shakti Standard Format (SSF) [32] is used for representing the analysis of a sentence. There are four columns in the SSF format. The first column is used for address, the second column is used for token, the third column is used for the category of the node, and the fourth column is used for other features. We can also annotate any other required linguistic information in this column using an attribute – value pair. The following example (example 4.2 is repeated here as example 4.43) illustrates how a sentence appears in SSF format after annotating the above linguistic information.

Ex- (4.43) raam ne mohan ko niili kitaab dii [28]
  ram Erg. mohan Dat. blue book gave
  ‘Ram gave a blue book to Mohan.’
The example taken above is represented in SSF format below, with morph, POS tags, and chunk information:

```
(( NP   <fs af=‘, , , , ’, name=‘NP’>
   raam NNP  <fs af= ‘root=raam, cat=n, gend=m, num=sg, pers=3, case=o’ name=‘raam’>
   ne PSP    <fs af= ‘root=ne, cat=psp, , , ’ name=‘ne’>
 ))
(( NP   <fs af=‘, , , , ’, name=‘NP2’>
   mohan NNP <fs af= ‘root=mohan, cat=n, gend=m, num=sg, pers=3, case=o’ name=‘mohan’>
   ko PSP    <fs af= ‘root=ko, cat=psp, , , ’ name=‘ko’>
 ))
(( NP   <fs af=‘, , , , ’, name=‘NP3’>
   niilii JJ  <fs af= ‘root=niilii, cat=adj, gend=f, , , , ’ name=‘niilii’>
   kitaab NN  <fs af= ‘root=kitaab, cat=n, gend=f, num=sg, pers=3, case=d’ name=‘kitaab’>
 ))
(( VGF  <fs name=‘VGF’ stype=‘declarative’ voicetype=‘active’>
   dii VM    <fs af= ‘root=de, cat=v, gend=f, num=any, pers=any, TAM/suffix=yaa, , ’ name=‘dii’>
 ))
```

As mentioned earlier, that the first column is address, we have not shown the address column here for the sake of simplicity.

Now we have again taken the same example and included the dependency relations along with morph, POS tags, and chunk information and represented it in the SSF format below:

```
(( NP   <fs af=‘, , , , ’, drel=‘karta (k1) : VGF’ name=‘NP’>
   raam NNP  <fs af= ‘root=raam, cat=n, gend=m, num=sg, pers=3, case=o’ name=‘raam’>
   ne PSP    <fs af= ‘root=ne, cat=psp, , , ’ name=‘ne’>
 ))
(( NP   <fs af=‘, , , , ’, drel=‘sampradana (k4) : VGF’ name=‘NP2’>
   mohan NNP <fs af= ‘root=mohan, cat=n, gend=m, num=sg, pers=3, case=o’ name=‘mohan’>
   ko PSP    <fs af= ‘root=ko, cat=psp, , , ’ name=‘ko’>
 ))
(( NP   <fs af=‘, , , , ’, drel=‘karma (k2) : VGF’ name=‘NP3’>
```
niilii JJ <fs af= ‘root=niilii, cat=adj, gend=f,’ name= ‘niilii’>
kitaab NN <fs af= ‘root=kitaab, cat=n, gend=f, num=sg, pers=3, case=d’ name= ‘kitaab’>
))
(() VGF <fs name= ‘VGF’ stype= ‘declarative’ voicetype= ‘active’>
dii VM <fs af= ‘root=dii, cat=v, gend=f, num=any, pers=any, TAM/suffix=yaa,’ name= ‘dii’>
))

Following is represented in the above annotation:

Morph Analysis → fs: feature structure; af: affected features; cat¹: category; n: noun; v: verb; adj: adjective; gend: gender; m: masculine; f: female; num: number; sg: singular; pers: person; o: oblique case; d: direct case; psp: postposition; TAM: Tense, Aspect, Modality

POS² Tags → NNP: Proper Noun; PSP: Postposition; JJ: Adjective; NN: Common Noun; VM: Main Verb.
(For the complete description of POS tags, see the guidelines: http://ltrc.iiti.ac.in/MachineTrans/publications/technicalReports/tr031/posguidelines.pdf) [33]

Chunks → NP: Noun Chunk (Noun Phrase); VGF: Finite Verb Group Chunk (Verb Phrase) (If a sentence is a simple sentence with finite verb then the verb chunk is tagged as VGF which is always the syntactic head of the sentence. In case of complex sentences also, VGF becomes the syntactic head of the complex sentence except sentences having coordination conjunctions joining two finite sentences where conjunction becomes the head of the complex sentence.)

(For the complete description of Chunk tags, see the guidelines: http://ltrc.iiti.ac.in/MachineTrans/publications/technicalReports/tr031/posguidelines.pdf) [33]

The above syntactic relations (dependency relations) are shown by means of features. A feature consists of attribute-value pairs. The attribute drel denotes ‘dependency relation’ and its value consists of two parts: (i) the type of dependency relation and (ii) the verb with which the relation holds. In the above example 4.43, the first word raam ‘Ram’ is related to the verb dii ‘gave’ by means of the karaka relation karta (k1). This is represented as ‘drel=k1: dii’. The type of sentence and the voice type (active/passive) of the verb are marked on the verb chunk. These are denoted by attributes stype and voicetype respectively. Sentence type of the above Hindi example, is declarative and its voice type is active. A dependency tree structure is formed by marking dependency relations in a sentence. The dependency tree structure for example 4.43 [28] is given below (same tree shown earlier for example 4.3):

¹ The feature cat in morph analysis denotes the lexical category of the word. This is more coarse-grained.
² POS tag is fine grained when compared to the cat feature of morph analysis. For example, POS tag captures the distinction between common noun and proper noun.
After discussing the dependency relations in this chapter, we discuss the major Hindi constructions and how they are represented in dependency structure in next chapter.

Figure 4.11 Dependency Tree structure-II