Empty Categories in a Hindi Treebank

Archna Bhatia[•], Rajesh Bhatt[†], Bhuvana Narasimhan[*], Martha Palmer[*], Owen Rambow[‡], Dipti Misra Sharma[◁], Michael Tepper[▽], Ashwini Vaidya[*], Fei Xia[▽]

[•] Univ. of Illinois at Urbana-Champaign, Urbana, IL 61801, USA, *bhatia@illinois.edu*
[*] Univ. of Colorado, Boulder, CO 80309, USA, {*narasimb,mpalmer,Ashwini.Vaidya*}*@colorado.edu*
[†] Univ of Massachusetts, Amherst, MA 01003, USA, *bhatt@linguist.umass.edu*
[‡] Columbia Univ, New York, NY 10115, USA, *rambow@ccls.columbia.edu*
[◁] Intl Institute of Information Technology, Hyderabad 500019, India, *dipti@iiit.ac.in*
[▽] Univ. of Washington, Seattle, WA 98195, USA, {*mtepper,fxia*}*@uw.edu*

Abstract

We are in the process of creating a multi-representational and multi-layered treebank for Hindi/Urdu (Palmer et al., 2009), which has three main layers: dependency structure, predicate-argument structure (PropBank), and phrase structure. This paper discusses an important issue in treebank design which is often neglected: the use of empty categories (ECs). All three levels of representation make use of ECs. We make a high-level distinction between two types of ECs, *trace* and *silent*, on the basis of whether they are postulated to mark displacement or not. Each type is further refined into several subtypes based on the underlying linguistic phenomena which the ECs are introduced to handle. This paper discusses the stages at which we add ECs to the Hindi/Urdu treebank and why. We investigate methodically the different types of ECs and their role in our syntactic and semantic representations. We also examine our decisions whether or not to coindex each type of ECs with other elements in the representation.

1. Introduction

We are in the process of creating a *multi-representational* and *multi-layered* treebank for Hindi/Urdu (Palmer et al., 2009). “Multi-layered” means that the treebank has different layers of representation: we represent both syntax and lexical predicate-argument structure. “Multi-representational” means that we use both dependency and phrase structure for syntactic representation. This paper is about an important issue in treebank design which is often neglected: the use of empty categories (ECs). ECs are a common representational device for treebanking, but they are often not specifically motivated, and may be unclear why an EC is used in a treebank to represent a syntactic fact, rather than some other representational device. The goal of this paper is to carefully explain and justify the types of ECs we use in our Hindi/Urdu treebank. We examine our choice of EC types and our decisions with respect to whether or not to coindex them with other elements in the representation. We also discuss the reasons why certain types of ECs are introduced at different stages during the annotation process. Our goal of automatic conversion from dependency structure to phrase structure places special emphasis on the role of empty arguments.

The paper is structured as follows. Section 2. provides an overview of the project, and Section 3. discusses two types of ECs in treebanking and motivates their use. In Section 4. we discuss in some detail the ECs introduced at our three levels of representation, Dependency Structure, PropBank, and Phrase Structure. This section also contains Table 2, which summarizes all types of ECs used in our project. We summarize our use of ECs in this project in Section 5., and then conclude.

2. An Overview of the Hindi/Urdu Treebank

This section provides a quick overview of the syntactic and semantic annotation choices we have made for the treebank, which has three main layers: Dependency structure, PropBank, and Phrase structure.

2.1. Dependency Structure (DS)

The Paninian grammatical model (Bharati et al., 1995; Begum et al., 2008) has been chosen as the basis of our dependency structure analysis. The sentence is treated as a series of modifier-modified relations which has a primary modified (generally the main verb). The appropriate syntactic cues (relation markers) help in identifying various relations. The relations are of two types: karaka (roles of various participants in an action, i.e., arguments, notated as k1-k6) and others (roles such as purpose, reason, and possession, which are captured using the relational concepts of the model, i.e., adjuncts).

2.2. Propbank (PB)

PropBanking is a semantic layer of annotation that adds predicate argument structures to syntactic representations (Palmer et al., 2005). For each verb, PropBank represents the information about the arguments that appear with the verb in its corresponding frame file. The arguments of the verbs are labelled using a small set of numbered arguments, e.g. Arg0, Arg1, Arg2, etc. Table 1 shows the frame file for the verb *pii* (*to drink*), which has two arguments: Arg0 and Arg1.

In the frame files for each verb, the numbered argument labels are associated with fine-grained verb-specific descriptions. Additionally, verb modifiers are annotated using functional tags such as ArgM-LOC, ArgM-TMP, ArgM-MNR. The PropBank framesets for Hindi will also include

'pii'	'to drink', Transitive
raam ne sharaab pii	
'Ram drank liquor'	
Arg0	drinker: *raam*
Arg1	liquid: *sharaab*

Table 1: A frame file for Hindi

mappings to the karaka (e.g., k1 and k2). Although PropBank annotation does not typically involve adding empty arguments to syntactic trees, in the case of Hindi-Urdu we have taken a somewhat different approach. We insert the core empty arguments of the verb (subject, object or indirect object) and assign them semantic role labels just as we do for the overt arguments. The resulting richer predicate argument structure on top of DS allows for high-quality conversion - an additional advantage of inserting empty arguments given our commitment to automatic conversion from dependency structure to phrase structure (Palmer et al., 2009). The information contained in the verb frame files can act as a valuable resource in allowing for recovery of the empty arguments before we annotate them using the semantic role labels.

2.3. Phrase Structure (PS)

The Phrase Structure guidelines are inspired by the Principles-and-Parameters methodology of Chomsky (1981). PS assumes a binary branching representation, where a minimal clause distinguishes at most two positions structurally (the core arguments). Any displacement of core arguments from their canonical positions is represented via traces. The displacement of other arguments is only represented if it crosses a clause boundary.

2.4. Overall Process

The treebank is created in three steps. The first step is the manual annotation of DS. In contrast with traditional DS annotation, we insert empty predicates (e.g., in the gapping construction) in DS when the empty predicates have dependents (arguments or adjuncts). The second step is PropBanking, which focuses on adding the lexical predicate-argument structure on the top of DS. Most empty arguments are added at this stage. The third step is the automatic creation of PS, which is done by a DS-to-PS conversion process that takes DS and PropBank as input and generates PS as output (Xia et al., 2009). For syntactic movement, the conversion process will insert traces and coindex them with the antecedents automatically.

In order to ensure successful conversion from DS/PropBank to PS, we are developing the three sets of guidelines for Hindi (DS, PB, and PS) simultaneously. While allowing DS and PS guidelines to be based on different, independently motivated principles of linguistic representation, we have been going through a comprehensive list of constructions in Hindi as a team, carefully exploring any issues that might impact the conversion process. In particular, during the guideline design phase, we make sure that DS and PB contain sufficient information to support the desired PS structure.

3. The Role of Empty Categories

3.1. Types of Empty Categories

We make a high-level distinction between different kinds of empty categories on the basis of whether they are postulated to mark displacement or not. Empty categories that mark displacement are called **traces**. In theories that use movement as a device to derive valid syntactic representations, traces are created as a result of movement. A trace is always co-indexed with another constituent in the sentence, and the interpretation is that the co-indexed constituent was, at a previous phase of the derivation, in the location of the trace (or, in some theories, at the same time).

All other empty categories are grouped under the label **silent**. These empty categories do not mark displacement; rather they represent undisplaced syntactic elements which happen to lack phonological realization. The label **silent** covers silent proforms as well as silent heads. Silent proforms are like other proforms (such as pronouns): they are co-referential with another element in the sentence or elsewhere in discourse, or implicit in the discourse situation. Examples include empty subject pronouns (*Are you hungry? Don't know.*), or empty clauses in Null Complement Anaphora (*A: Bill will be late again. B: I know.*). Silent heads resemble their non-silent counterparts. They differ from silent proforms in that they are not co-referential with other heads in the sentence or in the discourse context; they have their own reference. Examples include silent conjunctions, silent causative heads, and silent complementizers.

A different way of classifying empty categories emerges if we examine whether they can or must be co-indexed, and if they are what does the co-indexation indicate? As mentioned above, **traces** form a uniform class and always represent the movement relationship that forms them via co-indexation. The **silent** class is heterogenous with respect to co-indexation. The empty categories in this class are not a product of movement and co-indexation with elements of this class indicates a semantic dependency instead. However, we do not represent all semantic dependencies in the phrase structure; only those semantic dependencies which are completely determined by the syntax. Pronominal coreference (e.g. the relationship between *John* and *he* in *When John came back, he was upset*) is not fully determined by the syntax and for this reason it is not represented in the phrase structure. The referent of the silent PRO subject of an infinitival clause is sometimes fully determined by the syntax (*John wanted [PRO to dance]*), in which case it is represented via co-indexation; but in other cases the referent is left unspecified (*[PRO to dance] is fun*), and is not co-indexed. A similar case arises with silent VPs (*HEAD-VP*). In the case of gapping, the silent verbal projection is fully determined by the syntax and hence the silent VP is co-indexed with its antecedent. In the case of sluicing, however, the syntactic structure does not fully determine the antecedent of the silent VP and hence the semantic relationship between the silent VP and its antecedent is not represented via co-indexation in the phrase structure.

	Description	Type	Where Inserted	Label	Coindexed in PS?
1	Empty (Head of an) NP (ellipsis)	silent	DS	*HEAD-NP*	no
2	Empty (Head of a) VP (gapping, sluicing)	silent	DS	*HEAD-VP*	gapping:yes (structure sharing) sluicing: no
3	Empty subject with predicative adjective and "ki" complement clause	silent	DS	*pro*	no
4	Empty conjunction head	silent	DS	*CONJ*	no
5	Empty subject or object (regular pro-drop)	silent	PB	*pro*	no
6	Empty subject of a non-finite clause (control)	silent	PB	*PRO*	reference, but only if explicit
7	Empty relative pronoun	silent	PB	*RELPRO*	no
8	Trace of phrase undergoing extraposition (rightward movement)	trace	PS	*EXTR*	movement
9	Trace of phrase moving to case position in verbal projection	trace	PS	*CASE*	movement
10	Trace of a moved relative pronoun	trace	PS	*RELTR*	movement
11	Trace of scrambling (leftward movement)	trace	PS	*SCR*	movement
12	Trace of head moving to incorporate	trace	PS	*HEAD*	movement
13	Causative head	silent	PS	*CAUS*	no
14	Complementizer	silent	PS	*COMP*	no

Table 2: Empty category table. In column "Type", *trace* means that the empty category shows the original location of a moved element while *silent* indicates an unmoved category that just happens to lack an overt phonological realization. Column "Coindexed in PS?" shows whether the empty category has co-indexation with another element in the PS, and if so, whether the co-indexation is due to movement or reference.

3.2. Linguistic Motivation for Empty Categories

The primary reason why linguistic theories postulate ECs is that it allows for simpler descriptions. Often we find that sentences with certain empty elements have essentially the same properties (interpretation, case-marking, agreement) as the corresponding sentences where the element is overtly realized. In such cases, assuming that the putatively empty element is in fact realized by an empty category allows for simpler analyses; without such a postulation we would need two sets of rules - one where the element is overtly realized and one where it is not overtly realized. If the trees in a treebank are intended to follow the principles of some linguistic theory, then if the linguistic theory in question assumes ECs, so does the treebank.

Traces are the result of choosing a linguistic representation which includes movement, as we do for PS. While the result of movement (typically) reflects surface order (such as a fronted *wh*-element), the underlying position reflects other important information (such as licensing by a lexical predicate, i.e., lexical predicate-argument structure). Since we want to represent both surface order and underlying position, we use traces.

3.3. Empirical Motivation for Empty Categories

A different kind of motivation for postulating ECs comes from the demands of natural language processing, in particular information extraction, question-answering, and related semantic tasks. The more precise and detailed our predicate argument structures are (including empty categories), the more complete our event descriptions will be, and therefore the more effective our semantic processing techniques will be. As argued above, in certain representations, traces provide information as to the true governing head of the moved argument, which can also be used in appropriate semantic role labeling and semantic processing. Consider the difference between *[Which candidate]$_i$ do you expect t_i to win?* and *[Which prize]$_i$ do you expect to win t_i?*. The position of trace and the coindexation between trace and its antecedent indicate different readings of two sentences with very similar wordings.

4. Annotating Empty Categories

For the reasons just stated we have decided to include ECs in the treebank annotation. We use a very fine-grained representation of ECs, as summarized in Table 2. We discuss each stage of EC insertion below, emphasizing the criteria we use to determine at which layer ECs should be inserted.

4.1. Dependency Structure

In DS, only ECs that are required for completing a tree are inserted. ECs which would be leaf nodes in the DS tree are never included. The cases of ECs inserted in DS are listed in the first four rows of Table 2.

The first case is *Head-NP* for empty heads of NPs when the empty heads have dependents. One such example is in Ex (1), where the head of the NP "*piilii *Head-NP**" is empty (elided). In order to complete the tree, *Head-NP* is inserted in the DS to mark the empty head as shown in Figure 1.

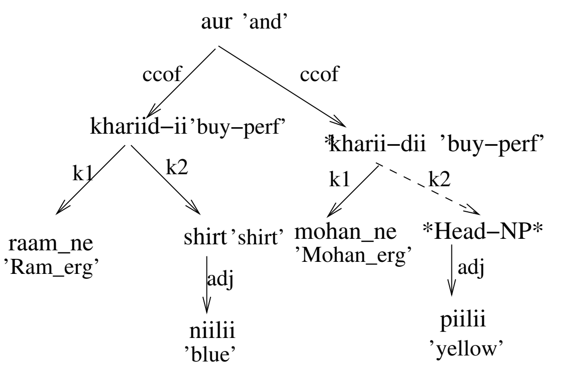


Figure 1: DS for the NP ellipsis example in Ex (1)

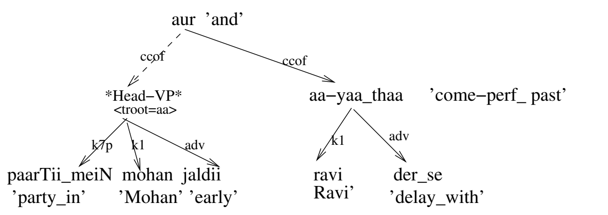


Figure 2: DS for the backward gapping example in Ex (2)

(1) Empty Head of an NP when the head has dependents:

raam ne niilii shirt khariid-ii aur mohan ne
Ram erg blue shirt buy-perf and Mohan erg
piilii *Head-NP* khariid-ii
yellow NULL buy-perf

'Ram bought the blue shirt and Mohan bought the yellow (one)'

The second case where an EC node is inserted in DS is that of backward gapping as illustrated in Ex (2). In order to provide a place for its arguments to attach to, an EC node *HEAD-VP* representing the empty verb is inserted in the DS and the arguments are then attached to it as shown in Figure 2.

(2) Empty Head of a VP as in backward gapping:

paarTii meiN mohan jaldii *HEAD-VP* aur ravi
party in Mohan early NULL and Ravi
der se aa-yaa thaa
delay with come-perf pst

'Mohan (came) early to the party and Ravi came late'

It should be noted that the *Head-NP* and the *Head-VP* empty categories are given distinct interpretations in the DS and the PS. The DS treats these as standing for heads of NP/VP while the PS treats these as silent nominal/verbal projections (NP/VP).

As mentioned above, empty arguments are normally not marked in DS since they tend to be leaf nodes in the DS. The only exception is when something else depends on an empty argument. An example is in Ex (3) (cf. Row 3 in Table 2), where a clause "*ki is saal monsoon der se aa-ii*" (that the monsoon arrived late this year) depends on the empty subject of the verb *hai* (is). An EC node representing the empty subject (*pro*) is inserted in the DS so that the head of the clause can depend on it.

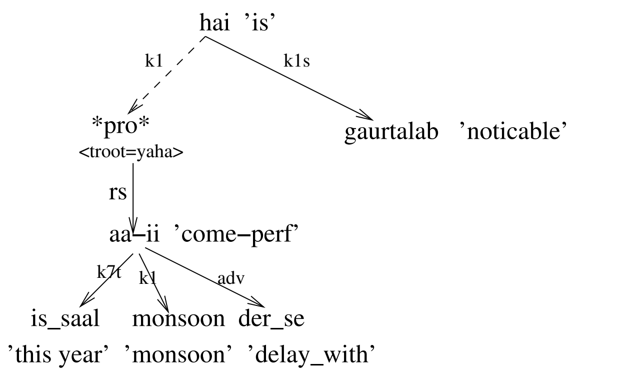


Figure 3: DS for the empty subject example in Ex (3)

(3) Empty subject with predicative adjective and "ki" complement clause:

pro gaurtalab hai ki is saal monsoon der
NULL noticable is that this year monsoon delay
se aa-ii
with come-perf

'It is to be noted that the monsoon arrived late this year'

Coordination is another case where DS inserts an EC. Coordination poses a problem for any dependency grammar based approach, and several alternative strategies exist for DS treatments of co-ordination (Nivre, 2005). At the DS level, the Hindi Treebank treats a conjunction as the root of a coordinate construction. However, sometimes a conjunction is not realized explicitly in a sentence in Hindi. In such cases, an EC node is posited to which its conjuncts are attached as illustrated in Figure 4 for the sentence in Ex (4).

(4) Empty conjunction head:

bacce baRe ho ga-ye haiN , *CONJ*
children old become GO-perf pres , NULL
kisii kii baat nahiiN maan-te
anyone gen advice not accept-imperf

'The children have grown big (and) do not listen to anyone'

4.2. Propbank

In the PB layer of the Hindi Treebank, we include the empty arguments of verbs. As mentioned above, the information about the semantic roles of arguments that appear with a verb is provided in corresponding frame files containing the verb's subcategorization information. This information is used to insert ECs for empty arguments.

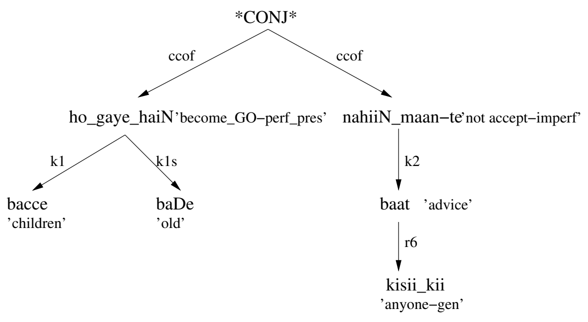


Figure 4: DS for the coordination example in Ex (4)

The set of ECs that are annotated at the PropBank stage include the following: core arguments of the verb (including subjects, direct and indirect objects) that are typically elided for discourse-pragmatic reasons (*pro* marked as *pro*), as well as empty subject arguments occurring in nonfinite complement and adjunct clauses (*PRO* marked as *PRO*) or empty arguments in relative clauses (labeled as *RELPRO*) that are controlled by antecedents within the same sentence. Examples of constructions with these different types of empty categories that are added at the PB layer are provided below.

As shown in example (5a), the subject argument of the transitive verb *paRh* 'read' can be elided, e.g. when it is recoverable from the prior discourse or situational context. The label *pro* is used not only to represent empty subjects as in (5a) but also objects, as in (5b):

(5) Examples of *pro*

a. Empty subject represented with *pro*:

pro kitaab paRh-egii
NULL book read-fut

'(She) will read the book'.

b. Empty object represented with *pro*:

kis ne darwaazaa khol-aa ? mohan ne
who erg door open-perf ? Mohan erg
pro khol-aa
NULL open-perf

'Who opened the door? Mohan opened (it)'.

Empty subjects that occur in nonfinite clauses are labeled as *PRO*. Example (6) is an example of an infinitival complement of the verb *chaah* 'want' with a syntactically empty subject that is controlled by the subject of the matrix verb:

(6) Empty subject of control verb shown by *PRO*:

mohan$_i$ ne [*PRO*$_i$ kitaab paRh-nii] caah-ii
Mohan erg NULL book read-Inf want-perf

'Mohan wanted to read the book.'

The category *RELPRO* in example (7) represents gaps in participial relative clauses that are used as prenominal modifiers of noun phrases:

(7) Gap in participial relative clause shown as *RELPRO*:

zyaadaatar [*RELPRO* kal khul-e]
most-of-the NULL yesterday open-perf
darvaaze
doors

'Most of the doors that opened yesterday'

For annotation, the arguments labeled *pro* will be inserted manually, using information from the context as well as the information contained in the verb frame files. The arguments labeled *PRO* (that are obligatorily non-overt) typically occur in a limited set of environments, e.g. in nonfinite complements of small number of control verbs as well as nonfinite adjunct clauses. Since the environments in which *PRO* occurs can be identified deterministically, the *PRO* labels will be inserted automatically during a preprocessing step. A similar solution is envisaged for the label *RELPRO*.

4.3. Phrase structure

The empty categories postulated by PS can be grouped into two classes: those created by movement (various kinds of traces) and those corresponding to unpronounced structure such as silent pronouns (*pro*, *PRO*), silent verbal ellipsis structures and silent heads.

For the reasons discussed in the preceding sections, empty arguments are represented using silent proforms. Empty subjects of non-finite clauses are represented by *PRO* and silent relative pronouns in participial relative clauses by *RELPRO*. Other empty arguments are represented by *pro*. Since theoretical assumptions do not force structural representation of adjuncts, we do not represent putatively empty adjuncts with silent pronouns unless this is forced by other considerations.

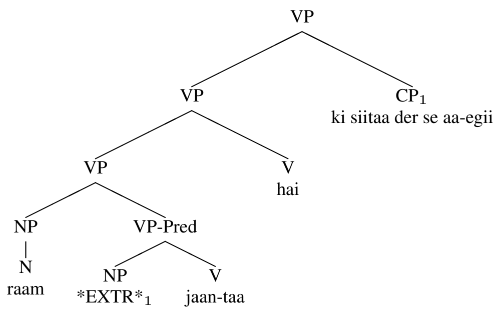
Silent pronouns are also used extensively in ellipsis environments, in particular NP-ellipsis and Null Complement Anaphora, both of which we treat as involving silent pronouns. The treatment of Gapping and Right Node Raising also makes reference to a silent verbal projection but unlike other cases of ellipsis (such as Null Complement Anaphora) the reference of the empty VP is completely determined by the structure. Therefore the empty VP (marked *HEAD-VP*) is always co-indexed with its antecedent.

The second class of empty categories ('traces') are used to mark displaced elements - they always bear an index that identifies their antecedent. The motivation for postulating such empty categories comes from the linguistic theory behind our particular PS treatment: not only must all arguments of a predicate be realized, they must be syntactically realized in their canonical positions (which represent the lexical predicate-argument structure of the governor). If the argument is displaced, then its canonical position is occupied by an empty category (its 'trace') and the displaced element is coindexed with the empty category. We make a number of distinctions between traces depending upon the direction and the motivation of the movement that produced the trace and the size of the object (phrase vs. head) whose movement produced the trace. We begin with traces of phrasal movement. Here we make a distinction between two kinds of traces: traces whose antecedent precedes them i.e. traces created by leftward movement and traces whose antecedent follows them i.e. traces created by rightward movement. The latter kind of trace is labelled *EXTR*.

(8) Trace of extraposed finite clause represented with *EXTR*:

raam *EXTR*$_1$ jaan-taa hai [ki siitaa der
Ram NULL know-hab pres that Sita delay
se aa-egii]$_1$
with come-Fut.FSg

‘Ram knows that Sita will come late.’

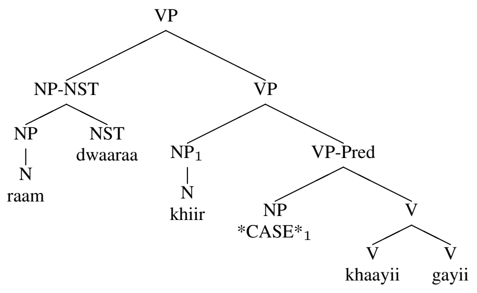


Even among traces created by leftward movement, traces differ in the motivation of the movement that created them. Traces created by movements driven by case motivations are marked *CASE*, traces created by movement of the relative phrase are marked *RELTR*, and finally traces created by all other kinds of leftward movement are marked *SCR*.

(9) Trace of case-related movement in a passive voice sentence represented with *CASE*:

raam dwaaraa khiir$_1$ *CASE*$_1$ khaa-yii
ram by rice-pudding NULL eat-perf
ga-yii
GO-perf

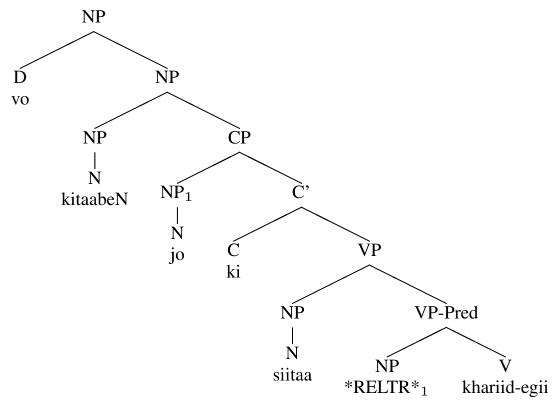
’Rice-pudding was eaten by Ram.’



(10) Trace of moved relative phrase represented by *RELTR*:

vo kitaabeN [jo$_1$ (ki) siitaa *RELTR*$_1$
those books Rel that Sita NULL
khariid-egii]
buy-fut

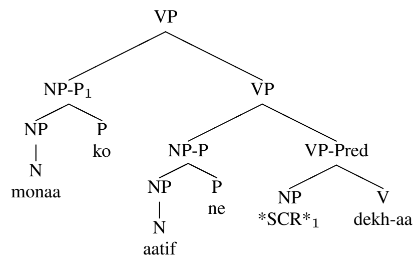
‘The books that Sita will buy.’



(11) Trace of scrambling-related movement represented with *SCR*:

[monaa ko]$_1$ aatif ne *SCR*$_1$ dekh-aa
Mona Acc Atif Erg NULL see-perf

‘Atif saw Mona./‘It was Atif who saw Mona.’

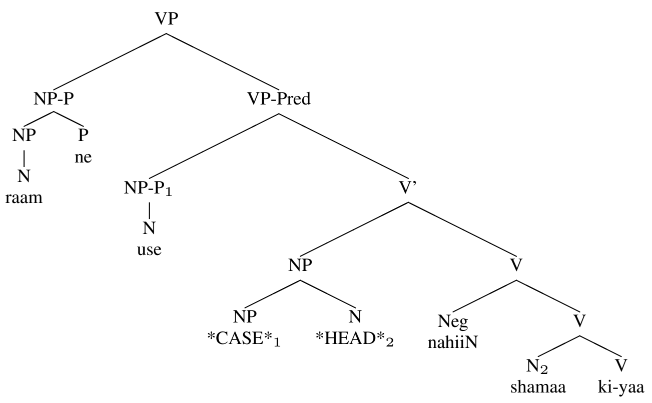


While most of the cases of traces in the treebank arise from the movement of phrases, certain structures involve movement of heads and the traces left behind by head-movement are indicated by *HEAD*.

(12) Trace of head-movement represented by *HEAD*:

raam ne use nahiiN *HEAD*$_1$ shamaa$_1$
Ram erg he.dat neg NULL forgiveness
ki-yaa
do-perf

‘Ram did not forgive him.’

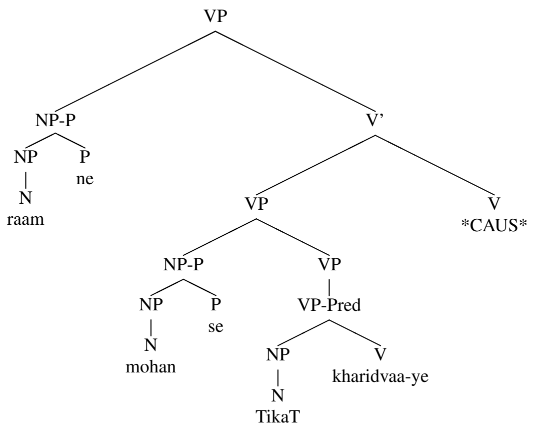


In addition to silent pronouns and traces, the phrase structure annotation introduces a silent causative head and a silent relative complementizer. These silent heads differ from traces and proform in that they do not depend upon another element of the same kind in the sentence or in the discourse context for their interpretation. The causative head introduces a causer argument and explicitly brings in causative semantics. There is a systematic relationship between the presence of the silent causative head in the phrase structure and the presence of a *-vaa* suffix on the verb that heads the VP complement of the causative head.

(13) Empty causative head represented by *CAUS*:

raam ne	mohan se	TikaT	kharidvaa-ye	*CAUS*
Ram erg	Mohan instr	ticket	buy-cause-perf	NULL

'Ram made Mohan buy the tickets.'



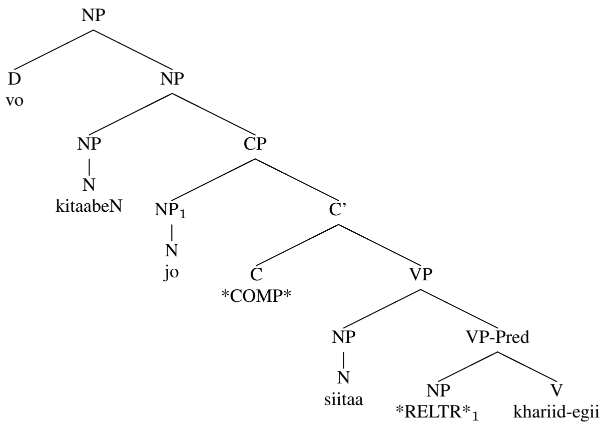
The *-vaa* causative alternation is highly productive in Hindi. With a few exceptions, any transitive or unergative verb in Hindi has a *-vaa* causative. We have chosen to represent this productivity in causative formation in the syntax. If the syntactic representations in our treebank allowed for a decompositional analysis of morphology, the *-vaa* suffix would occupy the position occupied by the silent causative head. But since we have not opted for a decompositional treatment, the silent causative head functions as a placeholder for the *-vaa* suffix on the causative verb; it functions as the head of a verbal projection which introduces the causer argument. Hindi also has other causative alternations that could in principle be represented using silent causative heads. We have chosen not to do so because these other alternations are neither as productive nor as morphologically transparent as the *-vaa* causative alternation.

Embedded finite clauses in Hindi-Urdu are introduced by the finite complementizer *ki*(*that*) (see Ex 8 and 10). However, there are also embedded finite clauses that do not involve a *ki*. In such cases, we postulate a silent complementizer to emphasize the structural and distributional parallels between finite clauses with and without *ki*.

(14) Empty complementizer head represented by *COMP*:

vo	kitaabeN	[jo$_1$	*COMP*	siitaa	*RELTR*$_1$	khariid-egii]
those	books	Rel	NULL	Sita	NULL	buy-fut

'The books that Sita will buy.'



5. Empty Categories, Annotation, and Information

Annotation is a process of adding linguistic information to an existing linguistic representation. Our project starts with the orthographic representation of a sentence. As summarized above, we have two manual annotation phases, and one automatic transformation which is not annotation. We discuss here in more detail the annotation process, especially with respect to ECs.

- **DS** starts with the orthographic representation of a sentence. DS is concerned only with representing the syntactic relations between the words (or chunks) that occur in the orthographic sentence, and adds exactly these relations. This is done in a manual annotation step which includes postagging and chunking as well as full syntactic structure (of course, preprocessing tools, including dependency parsers, are used to make the annotation easier). ECs are only added if this is necessary in order to form a tree while maintaining a consistent linguistic description.

- **PB** starts with the DS for a sentence. PB is concerned with the lexical predicate-argument structure of the verbs in a sentence, and adds empty arguments to the DS tree in order to make explicit the predicate-argument structure. This happens both via a preprocessing deterministic stage and manual annotation. Since PB always just extends the DS tree, all ECs introduced by DS are maintained.

- **PS**, unlike DS and PB, does not have a manual annotation stage. It is assumed that all the information necessary for the PS structure is already present, either explicitly or implicitly, in the DS/PB annotation. As discussed in (Xia et al., 2009), the DS-to-PS conversion process relies on being provided with a large number of examples of linguistic phenomena which illustrate in detail how to build the PS structure given the DS/PB input. Some changes, such as different arc labels, are fairly straightforward, and there are other commonalities. PS uses the ECs introduced by DS for the same reason (in order to create a tree), and it uses the ECs introduced by PB since it shares its interest in representing lexical predicate-argument structure. In many cases, such as movement, the PS structure is quite dramatically different from the DS/PB input and may involve the reordering of dependents. PS may also introduce additional ECs, such as traces. However, these alternative representations of movement that PS provides are still based on the underlying predicate-argument structure that is already captured in the DS/PB annotation.

Consider again Table 2. It is not a coincidence that all empty arguments are introduced at DS or PB (they represent new information being added manually), while all traces are added at PS (they are representational devices which can be deduced from other representational devices).

6. Conclusion

We have discussed the motivation and described the process for including empty categories, ECs, in the Hindi/Urdu Treebank. We have demonstrated that ECs can play an important role in producing rich linguistic representations but that clear definitions of their usages are needed. Our goal of creating a single treebank with multiple syntactic and semantic perspectives has caused us to carefully examine our intuitions concerning ECs, and to pinpoint precisely the most appropriate annotation layer for each type. We have described several different types of ECs, each one of which is used to represent different types of information in different types of representations: the ECs added during DS provide essential structural nodes; the ECs added during PB fill out subcategorization frames; and the EC's that the conversion process places in the PS are there to track the movement of displaced arguments.

7. Acknowledgment

This work is supported by the National Science Foundation CISE Collaborative Grants for A Multi-Representational and Multi-Layered Treebank for Hindi/Urdu, CNS-0751089, CNS-0751171, CNS-0751202, and CNS-0751213. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. References

Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai, and Rajeev Sangal. 2008. Dependency annotation scheme for indian languages. In *Proceedings of The Third International Joint Conference on Natural Language Processing (IJCNLP)*, Hyderabad, India.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. *Natural Language Processing – A Paninian Perspective*. Prentice-Hall of India.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. Technical report, Vaxjo University: School of Mathematics and Systems Engineering. MSI report 05133.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Martha Palmer, Rajesh Bhatt, Bhuvana Narasimhan, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, and Phrase Structure. In *Proceedings of ICON-2009: 7th International Conference on Natural Language Processing*, Hyderabad.

Fei Xia, Owen Rambow, Rajesh Bhatt, Martha Palmer, and Dipti Misra Sharma. 2009. Towards a multi-representational treebank. In *The 7th International Workshop on Treebanks and Linguistic Theories (TLT-7)*, Groningen, Netherlands.