8.1 Types of Null categories

Hindi-Urdu is a language that allows the speaker to freely omit arguments of the verb in discourse-pragmatically licensed contexts. For instance, one can say 'Ram drank liquor', but if 'Ram' has been talked about before, or is otherwise salient in the context, one could say 'drank liquor' without overtly mentioning the subject, 'Ram'. In a corpus, one come across many such sentences where the arguments of the verb are "missing" although they can be retrieved from the context.

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 empty categories corresponding to the core arguments of the verb including **subjects, direct and indirect objects** (so-called "little" *pro*, marked as *pro*), using the context to determine which sense of the verb is relevant.
- We also use the verb frame file to determine the number and semantic roles of the arguments that are not overtly realized (and that must be inserted).

For instance, in the following example, the subject argument (Arg0) of the transitive verb *paRh* 'read' can be elided, e.g. when it is recoverable from the prior discourse or situational context. In the second sentence, the object argument (Arg1) is missing.

1) **pro* kitaab paRh-egii*
   NULL book read-fut
   '(She) will read the book'.

2) *kis ne darwaazaa khol-aa? mohan ne *pro* khol-aa*
   *w*ho erg door open-perf? Mohan erg *pro* khol-aa
   'Who opened the door? Mohan opened (it)'.

In addition, three other kinds of obligatorily non-overt categories are inserted in PropBank:

- empty subject arguments occurring in nonfinite complement and adjunct clauses ("big" *PRO* marked as *PRO*);
- empty arguments in relative clauses (labeled as *RELPRO*)
- empty arguments in coordination and gapping constructions (labeled as *GAP-pro*) .

The contexts in which each of the empty arguments are inserted will depend mainly on **valency** considerations. This means that in an underlying fashion, verbs will have the capability of licensing arguments, but in the surface syntax they are unable to do so.

8.2 Null argument identification strategy

The process of identification of null arguments will take place concurrently with addition of PropBank labels. Hence, being alert to the various contexts in which an empty argument can occur is important. The following sections will describe the linguistic contexts in which the null categories can occur.
For the annotation of the null categories, we use a special drop-down menu in Jubilee. This shows the null argument label and the PropBank label together. The annotator has to choose the appropriate label for a particular instance of the annotation. Figure 1 below shows the drop-down menu:

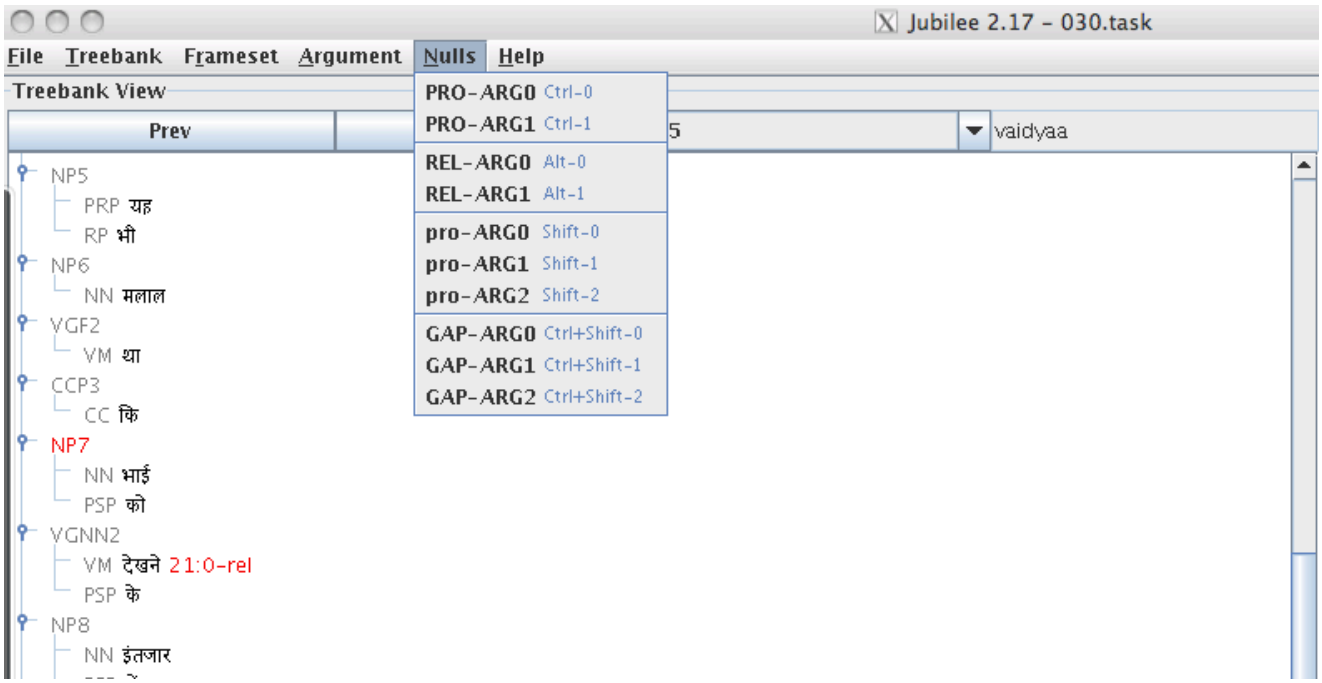

Figure 1: The null annotation drop-down menu in Jubilee

This method of annotation will identify the null argument *type* through the labels PRO, REL, pro and GAP. Simulatenously, it will allow for the annotation the correct PropBank label Arg0, Arg1 or Arg2 with each null type. Note that null arguments are always numbered (i.e. obligatory arguments)

The following sections will focus on the identification of the null argument types. This requires more attention to the syntactic information present in the tree, as it will provide important cues for the type of null argument required.

8.2.1 Specific constructions

In this section, the RELPRO and GAP-pro null categories will be discussed as the identification of these nulls depends largely on the specific construction that they occur in.

8.2.1.1 RELPRO

The RELPRO null category can occur in a specific construction; a participial relative clause. RELPRO stands for the empty relative pronoun in participial relative clauses in Hindi. Participial relatives are a relativization strategy where a modifying clause appears before the modified noun. This order of modifier-modified is typical of participial relatives. For example:

1. [Benca par **baithaa huaa**] ladkaa seb khaa rahaa hai

In this example, 'benca par baithaa huaa' is the participial relative clause modifying ladakaa. This means that [benca par baithaa huaa] is providing some additional information about the noun ladkaa. However, in the syntax, the noun ladkaa appears after this clause. In the dependency tree (shown below), the relationship between the noun and the relative clause is shown by a label **nmod__k1inv** OR **nmod__k2inv**.

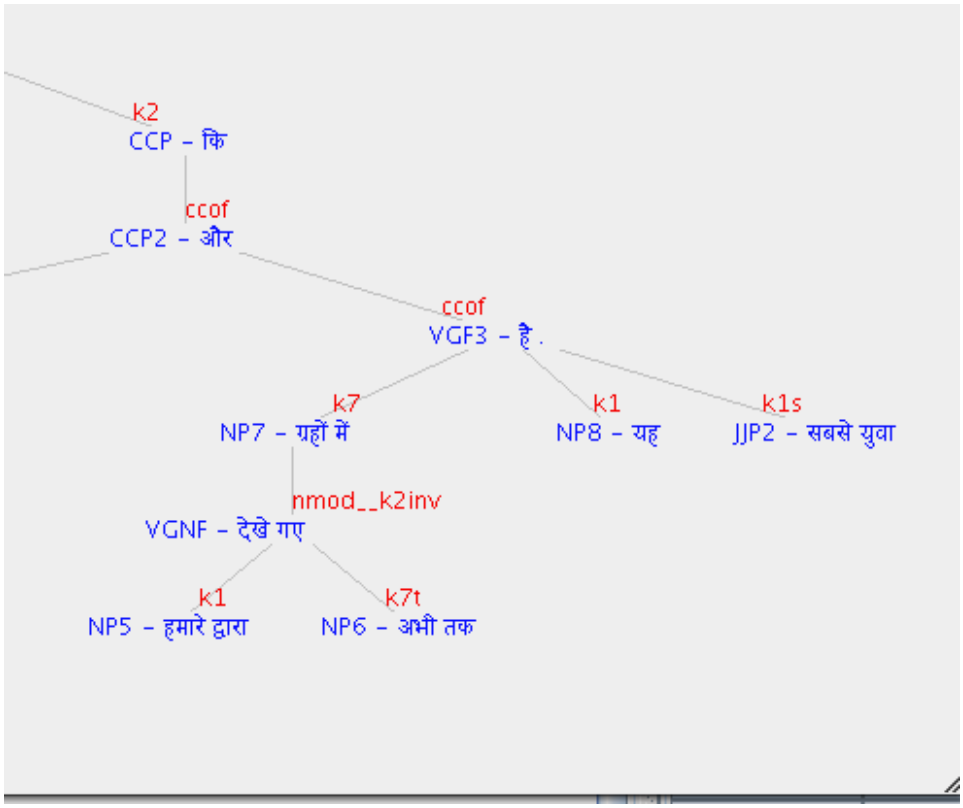

Fig2: Dependency tree showing the nmod__k2inv label

Figure 2 above shows the following example:
2. [hamaare dvaara abhi tak dekhe gaye] **grahon** mein yaha sabse yuvaa hai

In such a case, the argument 'grahon mein' is attached to the verb 'dekhe gaye' using an nmod__k2inv relation. In the presence of such a relation, we surmise that there is a missing relative pronoun in the clause [hamaare dvaara dekhe gaye]. Hence, we insert a RELPRO. The annotation will now look like the following:

2'. [hamaare dvaara$_{Arg0}$ RELPRO-$_{Arg1}$ abhi tak dekhe gaye$_{rel}$] grahon mein yaha sabse yuvaa hai

Note that RELPRO can have two options for a PropBank annotator: REL-ARG0 and REL-ARG1 (see Figure 1). Choosing the correct label will depend on the particular sentence, and the verb in this sentence.

The RELPRO argument can be inserted fairly deterministically in the presence of the nmod__k1inv or nmod__k2inv label. Sometimes, the label is nmod__pofinv. The RELPRO will also be inserted for the latter case.

8.2.1.2 GAP-pro

The GAP-pro annotation also requires the presence of a particular construction, in this case, the co-ordinate construction. A co-ordinate construction is two clauses joined with a conjunct (in Hindi, a conjunct could be aur 'and', lekin 'but' ya 'or' etc.) In a co-ordination construction, we have cases where arguments can be dropped.

3. [Mohan ne kitaab paRii] **aur** [so gayaa]

In the above example, the two clauses are joined by **aur.** mohan is the argument of padza 'read' and he is also the argument of so 'sleep'. It is possible to omit the

argument of ‘so’ because it is understood from the previous context. However, when we are annotating the arguments of ‘so’, we need to indicate that the doer of the action is actually an omitted argument. Figure 3 is an example of a tree from the corpus.

In this example, note that the conjunct ‘aur’ is the head of both verbs ‘kar’ and ‘dii’. The example in the tree reads as follows:

4. [udhar, John Carey ne apni haar swiikar kar lii] **aur** [Bush ko phone par badhaai dii]

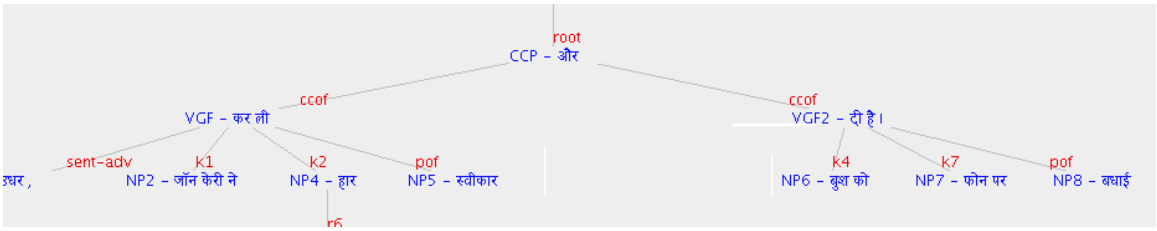

Fig 3: Gap-pro tree

Note that in the above example, John Carey is the same person who is wishing Bush on the phone. We will say that the argument ‘John Carey’ is omitted for the verb ‘dii’ and will be instead represented by a gap-pro. Further, this gap-pro will be co-indexed with the shared argument. (More details on co-reference will follow). The annotation with the inserted null argument will now look like this:

4’. [udhar, John Carey ne apni haar swiikar kar lii] **aur** [*gap-pro*$_{Arg0}$ Bush ko$_{Arg2}$ phone par$_{ArgM-MNS}$ badhaai dii]

The appearance of a co-ordination construction should immediately alert the annotator into looking for a gap-pro empty category. However, not ALL co-ordinate constructions have gap-pros. For example,

5. [mohan skuul jaane ki taiyari kar raha thaa] aur [uska bhai ro raha tha]
‘Mohan was ready to go to school and his brother was crying’

Here, the two clauses joined by 'aur' have nothing in common with each other, but are simply conjoined. In these cases, there is no sharing of a common argument. No empty category will be inserted.
In certain cases, the verb is also missing or gapped. However, PropBank does not insert the missing verb. In such cases, we will annotate as though the missing verb is already present.

6. Vaha ram ki kitaab hai aur vo mohan ki gap-pro NULL-VGF)

8.2.2 Nature of the verb

The remaining two empty categories can be identified with respect to the kind of verb that occurs in the sentence. The dropped subject/object/indirect object of a finite verb is the null argument 'pro' and the dropped subject or object of a non-finite complement or adjunct clause is 'PRO'.
We will first examine PRO cases and then move on to pro.

8.2.2.1 PRO

The PRO empty category is found mainly cases where annotation of a non-finite verb is required. Having first determined that the non finite verb is NOT a RELPRO, we can then go on to determine whether it can be a possible case of a PRO. The easy way to determine that a particular non-finite verb does not contain RELPRO is to check for the presence of the nmod__k1inv or nmod__k2inv label. If this is not present, then the next step is to determine whether the verb is non-finite. Such non-finite verbs have the labels VGNF or VGNN.

For example, Figure 4 below depicts the dependency tree for the following sentence:

7. PM [tabaahi kaa puura byoraa **jaananaa**] caahte hai

In this sentence, there are two verbs, but while caahte hai has the tag VGF2, ‘jaananaa’ gets VGNN.

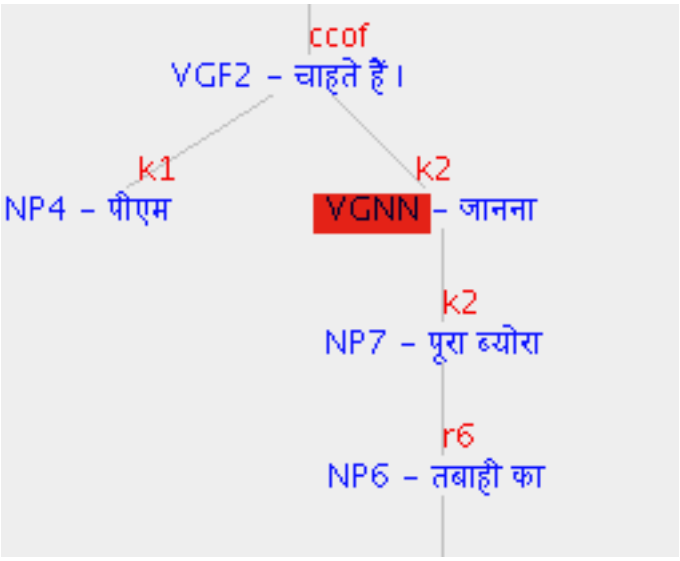

Fig 4: Tree for inserting PRO (005.10)009.10

The VGNN or VGNF tags show that the verb is non-finite, and in these cases, an argument is usually missing. This missing argument is a PRO.

If we look at the above sentence more carefully, the subject of ‘jaananaa’ is also the subject of the verb ‘caahnaa’ i.e. it is the argument ‘PM’. It is shared across both verbs, much like the case of gap-pro, which we looked at earlier. Unlike gap-pro, where there is clearly a co-ordination structure that we can reference, the clue for a PRO’s presence is the non-finite nature of the verb. Another useful clue is the verb frame file. Figure 5 shows that the verb ‘jaan’ requires a knower and thing known.

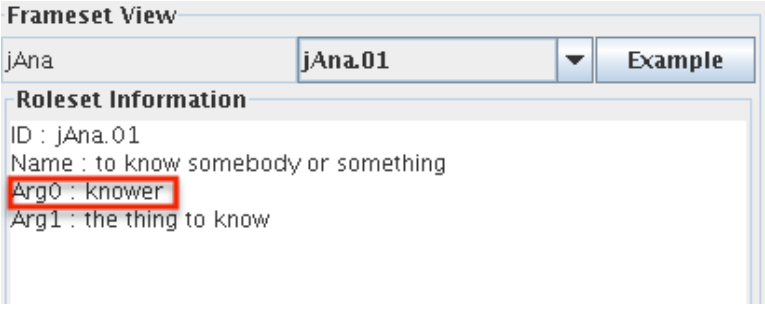

Figure 5: Frame file for the non-finite verb ‘jaana’

In the tree shown in Fig 4, the knower is clearly missing. The VGNN tag and the frame file information both point towards the insertion of a missing PRO. Having a shared argument is also a very useful pointer. The annotation for the above sentence will be as follows:

7'. PM [PRO_{Arg0} tabaahi kaa puura byoraa$_{ARG1}$ **jaananaa**] caahte hai

In some cases, there may be a missing subject which has no antecedent in the sentence. This means, it is not shared in the case of the example above. However, it is still the missing subject of a non-finite verb. This is a special type of PRO, called 'arbitrary PRO'. It is labeled in the same way as PRO, except it will not have a shared argument in another clause. Examples of these are seen in 8 and 9.

8. [PRO roz duudh **peena**] achcha hota hai
'To drink milk everyday is a good thing'

9. Do sau rupaya skuul ke shauchalay ke paas [PRO **giraa huaa**] milaa
Two hundred rupees school gen toilet gen near PRO fallen find
'Two hundred rupees fallen (on the ground) were found near the school toilet'

Example 9 is a depictive clause, explaining the state in which the money is found. The phrase 'giraa huaa' is explaining the state of the money, but as it is infinitival, we surmise that it has a missing PRO.

8.2.2.1 pro

The nature of the verb is also important to identify pro or 'small-pro'. In the case of pro, the verb is always finite, i.e. it will always have a tag 'VGF' and never VGNN or VGNF like the case of PRO. Moreover, the subject, object or indirect object may be dropped because it is understood via the context in which the sentence is written. For example, Figure 6 shows a case where the subject of the verb 'kharida' is missing.

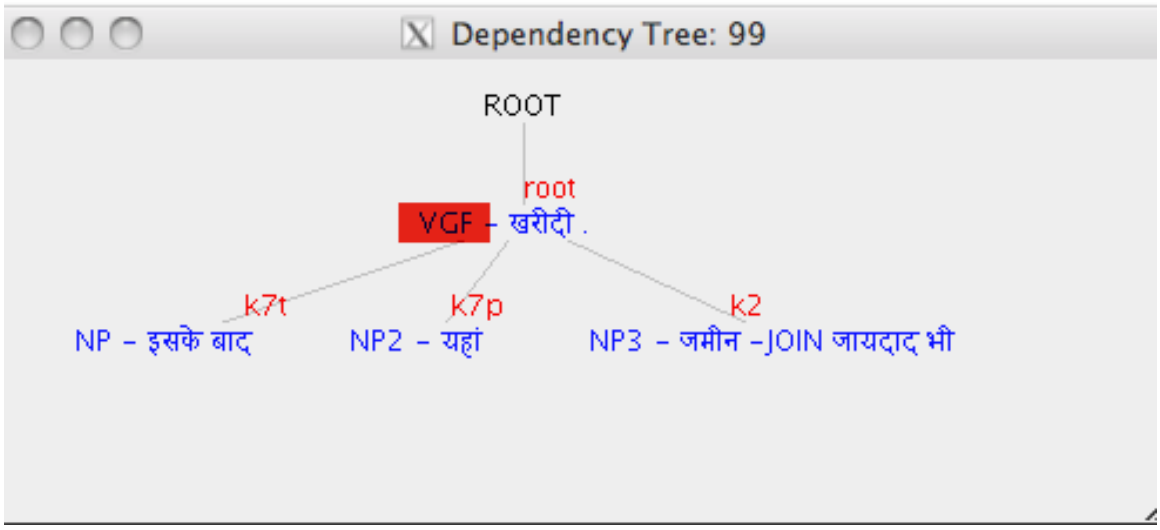

Figure 6: Tree for 'pro'

10. iske baad yahaan zameen-jaaydad bhi **kharidi**.

In this sentence, note that the verb kharidi is tagged as VGF, and it is missing a subject argument. Once again, if we refer to the frame file, it will give us a vital clue about the arguments of 'kharida'

Fig 7: Frame file for the verb 'kharida'

In Example 10, 'kharida' has a 'thing bought' (zameen-jaaydad) but it does not have a 'buyer'. This is the missing argument pro. The verb 'kharida' has the tag VGF and

its frame file denotes that an argument is missing. The annotation with the null inserted will now look like the following:

10'. pro$_{ARG0}$ iske baad$_{ARGM-TMP}$ yahaan$_{ARGM-LOC}$ zameen-jaaydad bhi$_{ARG1}$ **kharidi**.

The one context in which we do not insert pro is in the case of passive sentences. Refer to the wiki page for correct identification of passives:
http://resourcesforannotators.wikispaces.com/Passive_identification
For example, in 11, we have the passive verb 'bataaya gaya'

11. Munde ko is vishay ke baare mein bataa**ya** ga**yaa**
The passive cases do not require the insertion of 'pro'. However, note that in declarative sentences, we do need the insertion of 'pro' e.g. Example 12.

12. pro apne faayde ke liye, is updesh kaa paalan karein

8.3 Annotation of null categories

After having identified the correct type of null argument, each of these needs to be annotated using a different strategy. The actual annotation process for nulls will involve two types of actions:

- annotation on a shared argument: GAP-PRO, RELPRO, PRO
- annotation on the verb: pro

For any missing arguments that are coreferential with another argument in the same sentence (labeled **PRO**) or in one of the conjuncts of a conjoined sentence (labeled **GAP-PRO**, the *antecedent argument* will be double-annotated with the argument roles of each of the predicates with which it is associated. A similar strategy is used for **RELPRO,** although these cases don't have a shared argument, the label is simply placed on the modified noun.

For any missing arguments that are not co-referential with another argument that is 'local' (but which may be found in prior discourse), the *verb* will be annotated to indicate the missing argument(s) (labeled **pro** or the special cases of PRO without antecedents).

8.3.1 Annotation on shared argument

The correct null label should be chosen from the drop-down menu and then annotated, depending on whether the null type calls for shared annotation or annotation on the verb. The examples for shared annotation i.e. the GAP-pro and PRO cases are shown in Figure 8 and Figure 9.

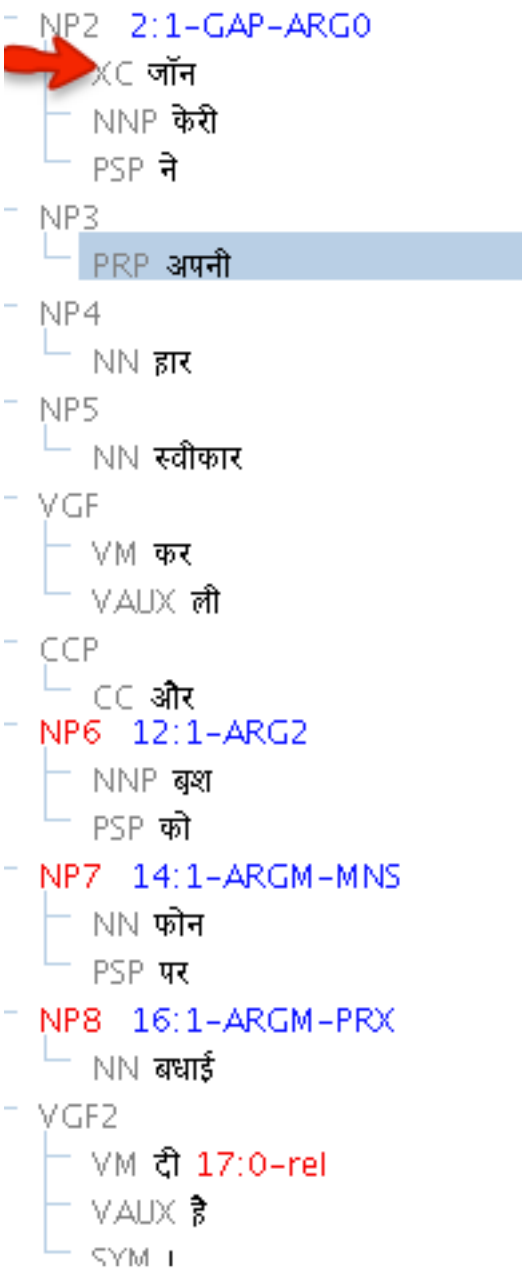

Figure 8: Example of shared annotation for GAP-pro

This is the same sentence as seen in Example 4. The sentence is repeated here:

4'. [udhar, John Carey ne apni haar swiikar kar lii] **aur** [*gap-pro*$_{Arg0}$ Bush ko$_{Arg2}$ phone par$_{ArgM-MNS}$ badhaai dii]

The verb being annotated is 'dii hai' but the GAP-ARG0 label is applied onto the shared argument in the previous clause, in this case it's John Carey. Note that John Carey will not be highlighted in red like the other arguments of the sentence. This is because it is a shared argument and belongs to the corresponding conjoined sentence with the verb 'swiikar kar'.

The same strategy is applied to the cases where PRO is inserted and it has a shared argument. Figure 9 shows the annotation.

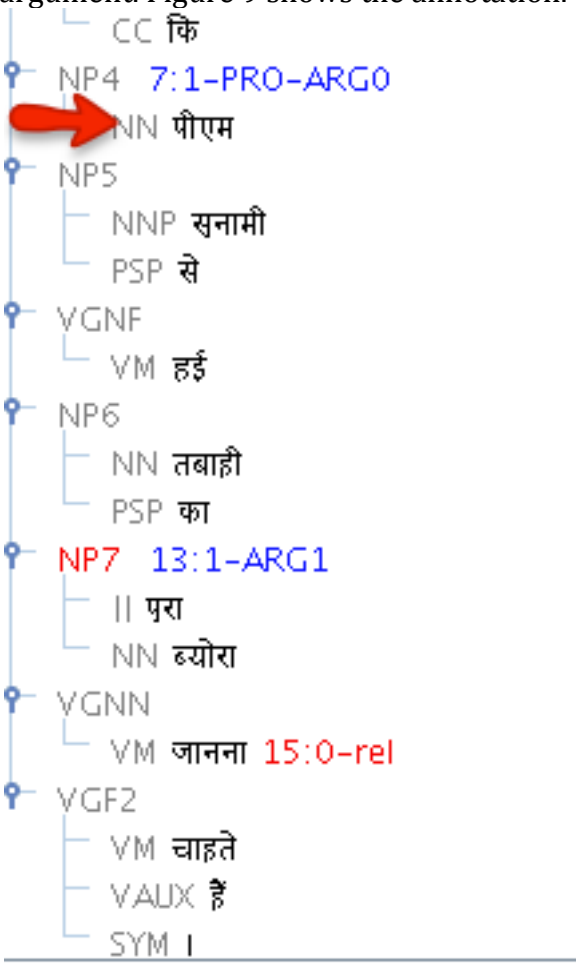

Figure 9: Null annotation for PRO 'shared' cases

Again, this is the same example 7 sentence, reproduced below:

7'. PM [PRO_{Arg0} tabaahi kaa puura byoraa$_{ARG1}$ **jaananaa**] caahte hai

The argument 'PM' is shared across both verbs and hence PM gets the label PRO-ARG0. Similar to GAP-pro, it is the argument from another clause and won't be highlighted in the tree.

In the case of RELPRO, a strategy of double annotation is carried out, but the

arguments are not shared in the sense of GAP-pro and PRO. The noun that is modified acts as a placeholder for the empty category.
The example below shows the same sentence as Example 2, reproduced below:

2'. [hamaare dvaara$_{Arg0}$ RELPRO-$_{Arg1}$ abhi tak dekhe gaye$_{rel}$] grahon mein yaha sabse yuvaa hai

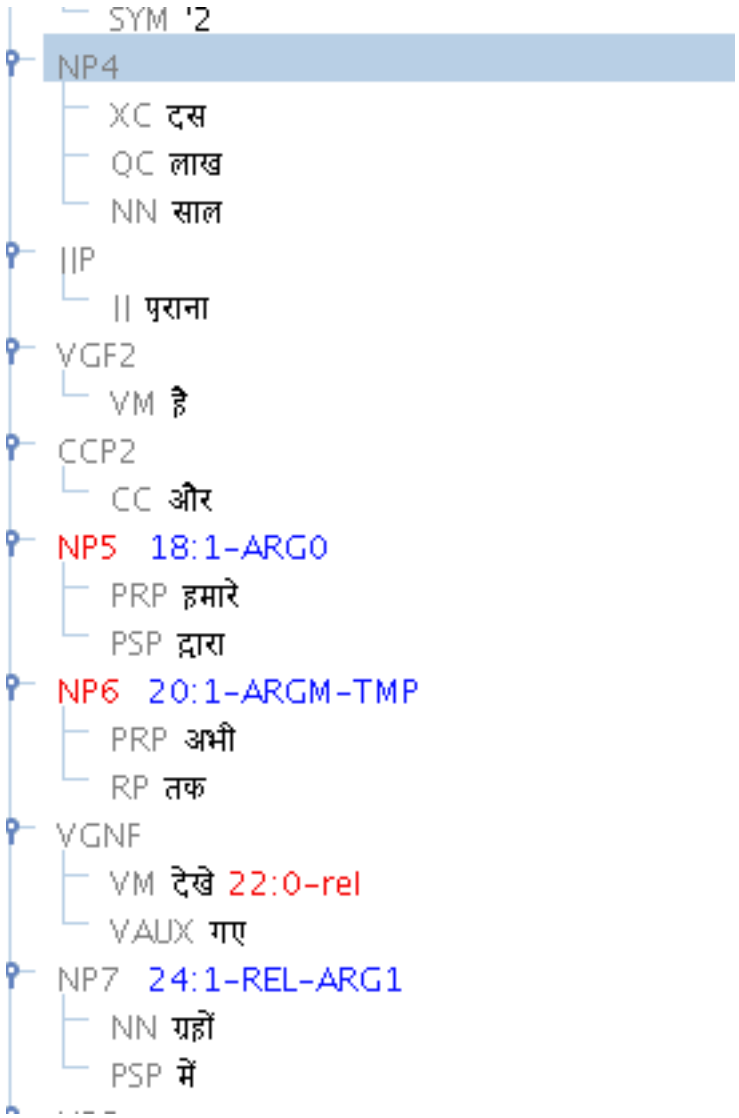

Figure 10: RELPRO annotation on the noun

In this case, 'grahon mein' is the modified noun and gets the label REL-ARG1.

8.3.1 Annotation on the verb

Single annotation is carried out for those cases where there is no shared argument, and hence there is no actual placeholder for the null argument. This takes place for the pro type of null argument and also for the special cases of PRO that do not have antecedents. Figure 11 shows the single annotation strategy for pro, where the verb's node gets the pro-ARG0 label.

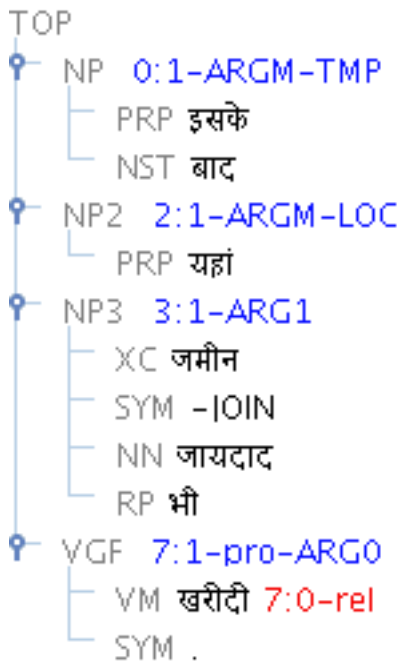

Figure 11: Single annotation of pro on the verb

The figure above shows the annotation *on the verb* for the null argument pro. This is the same as example 10, below.

10'. pro$_{ARG0}$ iske baad$_{ARGM-TMP}$ yahaan$_{ARGM-LOC}$ zameen-jaaydad bhi$_{ARG1}$ **kharidi**.