4.1 Introduction

EnConverter is a language analysis system that converts a source language sentence into UNL expression by using lexical information and EnConversion analysis rules of source language. In this work, a Punjabi EnConverter that takes Punjabi sentence as input and produces its equivalent UNL expression as output has been developed. The proposed system uses Punjabi-UW dictionary for corresponding universal words; and Punjabi analysis rules for morphological, syntactic and semantic processing of input sentence. Architecture of the Punjabi-UNL EnConverter and its implementation has been discussed in this chapter.

4.2 Working of Punjabi EnConverter

Punjabi EnConverter processes a given input sentence from left to right. It uses two windows, namely, analysis window and condition window (Dhanabalan et al., 2002; Dave and Bhattacharyya, 2001) while processing a given Punjabi sentence. The currently focused windows, i.e., analysis windows are circumscribed by condition windows as shown in Figure 4.1.

![Figure 4.1: Analysis and condition windows of Punjabi EnConverter](image-url)

EnConverter

Word Dictionary

EnConversion Rules

Node-list

\[
\begin{array}{cccccccc}
 n_{i-2} & n_{i-1} & n_i & n_{i+1} & n_{i+2} & n_{i+3} \\
\end{array}
\]
In Figure 4.1, ‘A’ indicates an analysis window, ‘C’ indicates a condition window, and ‘ni’ indicates an analysis node. The EnConverter analyzes the sentence using word dictionary and EnConverter analysis rules. The database of these EnConversion rules is created on the basis of morphological, syntactic and semantic information of Punjabi language as recommended by Uchida (1987), Dave and Bhattacharyya (2001), Dey and Bhattacharyya (2005).

4.3 Format of EnConversion analysis rules

EnConversion analysis rules have been designed on the guidelines mentioned in UNL EnConverter Specifications (UNL, 2000). The rule format used in designing of system is given in (4.1).

\{COND1:ACTION1:REL1\} \{COND2:ACTION2:REL2\} \ldots (4.1)

Here,
• <COND1> indicates Condition1 and contains the lexical and semantic attributes of left analysis window.
• <COND2> indicates Condition2 and contains the lexical and semantic attributes of right analysis window.
• <ACTION1> and <ACTION2> are used to indicate the actions performed if the corresponding condition is true.
• The <REL1> and <REL2> fields indicate possible relation between two analysis windows.

The EnConversion analysis rules are categorized into five types, namely, left composition rules (+), right composition rules (-), left modification rules (<), right modification rules (>) and attribute changing rules (:). A brief description of each type is given below.

4.3.1 Left composition rule (+)

A left composition rule is preceded by ‘+’ sign in the rule base. During its processing, the right node of analysis window is combined to left node of analysis window to make a composite node and the attributes of left analysis node are inherited for further processing, \textit{i.e.}, the headwords of left and right nodes of analysis windows are combined into a composite node and the original left and right nodes of analysis windows are replaced with the composite node in the node-list. The position of new composite node is on the left analysis window.
4.3.2 Right composition rule (-)
A right composition rule is preceded by ‘-’ sign in the rule base. During the processing of this rule, the left node of analysis window is combined with the right node of analysis window to make a composite node and the attributes of right analysis node are inherited for further processing. Here, headwords of the left and right nodes of analysis windows are combined into a composite node and the original left and right nodes of analysis windows are replaced with the composite node in the node-list. The position of the new composite node is on the right analysis window.

4.3.3 Left modification rule (<)
A left modification rule is preceded by ‘<’ sign in the rule base. This rule is applicable when right node of analysis window modifies the left node of analysis window. In this case, the right node of analysis window becomes the modifier of the left node of analysis window and it results into deletion of right node of analysis window from the node-list, while the left node of analysis window becomes the head of this partial syntactic tree and remains in the node-list.

4.3.4 Right modification rule (>)
This rule is preceded by ‘>’ sign in the rule base. This rule is applicable when left node of analysis window modifies the right node of analysis window. In this case, the left node of analysis window becomes the modifier of the right node of analysis window that results into deletion of left node of analysis window from the node-list, while the right node of analysis window becomes the head of this partial syntactic tree and remains in the node-list.

4.3.5 Attribute changing rule (:)
An attribute changing rule is preceded by ‘:’ sign in the rule base. The attribute changing rule is used to add or delete attributes from a particular node. When the attributes are to be added to a particular node then they are preceded by ‘+’ sign and when the attributes are to be deleted from a node then they are preceded by ‘-’ sign.

The syntactic analysis of input sentence is carried out by left composition rule (+), right composition rule (-), left modification rule (<) and right modification rule (>). These rules generate a sub-syntactic tree based on the conditions of analysis windows. The semantic analysis is carried out by the relation field of left modification rule (<) and right
modification rule (>). It results into a binary UNL relation between UWs of nodes of the analysis windows (Shah et al., 2000).

The working of EnConversion analysis rules is illustrated below with an example rule given in (4.2).

\[
>\{N, \text{INS}:null:ins\}\{V:+\text{INSRES}:null\} \quad \text{...(4.2)}
\]

The rule given in (4.2) is proceeded by ‘>’ sign. This means that it is a right modification rule, which results into the deletion of left node from the node-list and the attributes of right analysis window are considered for further processing. Here, ‘N, INS’ (noun, instrument) in the condition field of left analysis window and ‘V’ (verb) in the condition field of right analysis window, mean that these attributes are required to be present on the respective analysis windows for the firing of this rule. As action field of left analysis window contains ‘null’, so no action is required to be performed on the left analysis window. The action field of right analysis window contains ‘+INSRES’, this results into addition of ‘INSRES’ attribute to the lexical semantic attribute list of node on right analysis window. Since, the relation field of left analysis window contains ‘ins’ and the relation field of right analysis window contains ‘null’, it results into the resolution of ‘ins’ relation between left and right analysis windows. Here, right analysis node acts as the parent while left analysis node acts as child of this relation.

4.4 Punjabi shallow parser

Punjabi EnConverter developed in this work uses Punjabi shallow parser (developed by Consortium of Institutions headed by IIIT Hyderabad, India) for processing input Punjabi sentence. This parser performs the tasks of tokenization, morph analysis, part-of-speech tagging and chunking for the processing of an input sentence. It produces final output by picking most appropriate morph with ‘head’ and ‘vibhakti’ computation. It also has the provision of using the output of each intermediate stage. It generates the output in ‘Shakti’ format. ‘Shakti’ format uses a common representation for the operation of all modules (Bharati and Sangal, 1993; Bharati et al., 2007).

The working of Punjabi shallow parser has now been illustrated with an example sentence given in (4.3).

Punjabi Example sentence: ਛੋਟੇ ਬੱਚਾਂ ਨੇ ਵਕਤਾਬ ਪੜਹੀ।  

Transliterated example sentence: chōṭē baccīāṁ nē kitāb paṭhī.  

…(4.3)
Equivalent English sentence: Little children read the book.
The output of Punjabi shallow parser for this example sentence at each intermediate stage has been explained in subsequent sub-sections.

**4.4.1 Tokenizer**
A token is an instance of a sequence of characters in a sentence that are grouped together as a useful semantic unit for processing. The tokenizer converts a sentence into word level tokens consisting of words, punctuation marks, and other symbols. The output of tokenizer for the example sentence given in (4.3) is shown in (4.4).

**Tokenizer:**

```xml
<Sentence id="1">
  1 ਛੋਟੇ unk
  2 ਬੱਵਚਆਂ unk
  3 ਨੇ unk
  4 ਵਕਤਾਬ unk
  5 ਪੌਛੀ unk
</Sentence>
```

At this stage there is no part-of-speech information resolved for the tokens, so each token has a ‘unk’ unknown token attribute.

**4.4.2 Morph analyzer**
The morphological analyzer identifies the root and the grammatical features of the word. The output of morph analyzer for example sentence is shown in (4.5).

**Morph analyzer:**

```xml
<Sentence id="1">
  1 ਛੋਟੇ unk <fs af='ਛੋਟਾ,adj,m,sg,,o,'/>
  2 ਬੱਚਾ ਆਂ unk <fs af='ਬੱਚਾ,n,m,pl,3,o,,'/>
  3 ਨੇ unk <fs af='ਨੇ,psp,,,,d,,,'|<fs af='ਆਂ,v,any,pl,2,,,'/>
  4 ਵਕਤਾਬ unk <fs af='ਵਕਤਾਬ,n,f,sg,3,d,,,'/>
</Sentence>
```
Here, ‘fs’ is the feature structure which contains grammatical features of each word and ‘af’ is a composite attribute which consists of seven attributes, namely, root; lexical category; gender; number; person; case; tense, aspect and modality (TAM) information. In case, no value is given for a particular attribute the field is left blank. As shown in (4.5), छोटे chōṭē has a root word छोटा chōṭā; lexical category as ‘adj’ (adjective); gender as ‘m’ (male); number as ‘sg’ (singular) or ‘pl’ (plural), i.e., correct number is not resolved at this stage, so, both singular and plural forms are shown in the feature structure; person is ‘not applicable’ for this token; case is ‘o’ (oblique) and TAM is ‘not applicable’ for this token.

4.4.3 Part-of-speech tagger

Part-of-speech tagging is the process of assigning a part-of-speech to each token in the sentence. It helps in analyzing the role of each constituent in a sentence. The output of part-of-speech tagger for example sentence given in (4.3) is shown in (4.6).

POS tagger:

```
<Sentence id="1">
1  छोटे JJ <fs af='छोटा,adj,m,sg,.'>
   <fs af='�ोटा,adj,m,pl,.'>
2  बच्चाएँ NN <fs af='बच्चा,n,m,pl,3,.'>
3  ने PSP <fs af='ने,psp,3,d,.'>|<fs af='नो,v,any,pl,2,.'>
4  बिउति NN <fs af='बिउति,n,f,sg,3,d,.'>
5  पढ़ी VM <fs af='पढ़ा,v,f,sg,any,.'>
</Sentence>
```

As shown in (4.6), छोटे chōṭe ‘little’ is an adjective with symbol ‘JJ’, बच्चाएँ bacciāṃ ‘children’ and बिउति kitāb ‘book’ are nouns with symbol ‘NN’, ने nē is postposition with symbol ‘PSP’ and पढ़ी paṭhī ‘read’ is the main verb with symbol ‘VM’ (Bharati et al., 2006).
4.4.4 Chunker

Chunking involves identifying noun phrases, verb groups, adjective phrase, and adverb phrase in a sentence. It involves identifying the boundary of chunks and the label. The output of chunker for example sentence given in (4.3) is shown in (4.7).

Chunker:
--------

<Sentence id="1">
1  ((  NP
1.1 ਛੋਟੇ JJ  <fs af='ਛੋਟਾ,adj,m,sg,,o,'>|<fs af='ਛੋਟਾ,adj,m,pl,,o,'>
1.2 ਬੱਚਆਂ NN  <fs af='ਬੱਚा,n,m,pl,3,o,'>
1.3 ਨੇ PSP  <fs af='ਨੇ,psp,,,,d,,'>|<fs af='ਆਂ,v,any,pl,2,,,'>
))
2  ((  NP
2.1 ਵਕਤਾਬ NN  <fs af='ਵਕਤਾਬ,n,f,sg,3,d,'>
))
3  ((  VGF
3.1 ਪੜਹੀ VM  <fs af='ਪੜਹ,v,f,sg,any,,ਇਆ,ਇਆ'">
))
</Sentence>  ...(4.7)

As shown in (4.7), the example sentence has three chunks, ‘ਛੋਟੇ ਬੱਚਆਂ ਨੇ’ ‘chōṭē bacciāṁ nē’ ‘little children’ and ਵਕਤਾਬ kitāb ‘book’ as noun phrase ‘NP’ chunks and ਪੜਹੀ pathī ‘read’ as finite verb chunk ‘VGF’.

4.4.5 Pruning

Pruning stage identifies the most appropriate feature structure out of different possible structures. Pruning involves two types of sub-pruning, namely, morph pruning and pick one morph. Morph pruning takes those feature structures where the lexical category value matches with the category value. If there is no feature structure whose lexical category matches with the category, then all the feature structures are retained and a new attribute ‘NM’ indicating not matched is added to corresponding feature structure.
Pick-one-morph picks only one feature structure based on selection definition given to it. By default, it will pick the first feature structure. For the example sentence given in (4.3), the output of pick-one-morph of pruning stage is given in (4.8).

The proposed Punjabi EnConverter makes use of output of pruning stage for further processing of an input sentence.

4.4.6 ‘head’ computation

At this stage a child node is identified as ‘head’ of the chunk. The chunk node inherits the properties of the ‘head’ child. A new attribute ‘head’ is added to the feature structure of the chunk node whose value is the name-string assigned to the ‘head’ child. The output of ‘head’ computation stage of the example sentence given in (4.3) is shown in (4.9).
2.1 विरुध ं NN <fs af='विरुध्न, n,f,sg,3,d,,' name='विरुध्न'>

3 (( VGF <fs af='पढ़, v,f,sg,any,,रिशात्र,रिखर' head='पढ़ी'>

3.1 पढ़ी VM <fs af='पढ़, v,f,sg,any,,रिशात्र,रिखर' name='पढ़ी'>

4.4.7 ‘vibhakti’ computation

In this stage function words are grouped with the content words based on local information. This module computes the case/TAM features of noun/verb chunks and adds them to feature structure. The ‘vibhakti’ computation output for the example sentence is given in (4.10).

As shown in (4.10), in chunk-1, function word नेत nē is grouped with the ‘head’ word बच्चा

‘children’ with ‘vibhakti’ ‘vib2_3’ and in chunk-3, verb modifier इया iā is grouped with ‘head’ word पढ़ी pathī ‘read’.

In the next section, the architecture of the proposed Punjabi-UNL EnConverter has been presented.
4.5 Punjabi EnConverter architecture

The architecture of Punjabi EnConverter can be divided into seven phases, including an optional phase. It consists of the tasks of processing of input Punjabi sentence by Punjabi shallow parser, creation of linked list of nodes on the basis of output of shallow parser, extraction of UWs and generation of UNL expression for the input sentence. The phases in proposed Punjabi EnConverter are as follows.

i) Parser phase (to parse the input sentence with Punjabi shallow parser)

ii) Linked list creation phase

iii) Universal Word lookup phase

iv) Case marker lookup phase

v) Unknown word handling phase

vi) User interaction phase (this phase is optional)

vii) UNL generation phase

Figure 4.2 describes the flow chart illustrating working of Punjabi EnConverter. In the next sub-sections, these phases are explained, in brief.

4.5.1 Parser phase

Punjabi EnConverter uses Punjabi shallow parser discussed in Section 4.4 for parsing an input Punjabi sentence to produce the intermediate outputs of tokenizer, morph analyzer, part-of-speech tagger, chunker, pruning, ‘head’ and ‘vibhakti’ computation.

4.5.2 Linked list creation phase

In this phase, Punjabi EnConverter constructs a linked list of nodes. This linked list is constructed on the basis of information generated by the Punjabi shallow parser, Punjabi-UW dictionary and verb-modifier database. For the process of creation of linked list, the output of pruning stage of Punjabi shallow parser has been used. Each root word of the token and verb modifiers of the main verb act as the candidates for the node.

For each root word, the words obtained by combining it with root word of next consecutive tokens are searched in Punjabi-UW dictionary and verb-modifier database, so that the largest token can be formed on the basis of root words stored in the Punjabi-UW dictionary (e.g., ‘ਸਯ ਕਤ ਰਾਸ਼ਟਰ ਅਮਰੀਕਾ’ ‘sayukat rāṣṭar amrīkā’ ‘United States of America’) or on the basis of groups of words that modify the root verb (e.g., ‘ਰਹੀ ਹੈ’, ‘rahī hai’).
Figure 4.2: Flowchart for working of Punjabi EnConverter

* Developed by Consortium of Institutions headed by IIIT Hyderabad, India.
If a token formed by the concatenation of consecutive root words is found as a single entry in Punjabi-UW dictionary or in verb-modifier database, then that group of words is considered as a single token and stored as a node in the linked list, otherwise, each root word of the token is considered as a single token and stored as a node in the linked list. A node in the linked list has Punjabi root word attribute, Universal Word attribute, Part-of-Speech (POS) information attribute, and a list of lexical and semantic attributes. The structure of a node is given in Figure 4.3.

![Figure 4.3: Structure of a node](image)

In Figure 4.2, ‘D1’, ‘D2’ and ‘D3’ are the lexical and semantic attributes of the node. For the example sentence given in (4.3), the output of pruning stage given in (4.8) is used to create the linked list. The candidates for the nodes of linked list are ਛੋਟਾ ‘little’, ਬੱਚਾ ‘child’, ਨੇ ‘nē’, ਵਕਤਾਬ ‘book’, ਪੜਹ ‘read’, ਇਆ ‘iā’. For the first node, words ਛੋਟਾ, ‘ਛੋਟਾ ਬੱਚਾ’, ‘ਛੋਟਾ ਬੱਚਾ ਨੇ’, ‘ਛੋਟਾ ਬੱਚਾ ਨੇ ਵਕਤਾਬ’ are searched in Punjabi-UW dictionary and verb-modifier database. In this case, a node will be created for the token, ਛੋਟਾ ‘little’, in the linked list, because it has an entry in Punjabi-UW dictionary. Similarly, the nodes are created for ਬੱਚਾ ‘child’, ਨੇ ‘nē’, ਵਕਤਾਬ ‘book’, ਪੜਹ ‘read’ and ਇਆ ‘iā’. Thus, example sentence given in (4.3), results into creation of linked list of six nodes, namely, ਛੋਟਾ ‘little’, ਬੱਚਾ ‘child’, ਨੇ ‘nē’, ਵਕਤਾਬ ‘book’, ਪੜਹ ‘read’ and ਇਆ ‘iā’.

### 4.5.3 Universal Word lookup phase

In this phase, Punjabi-UW dictionary is used for mapping of Punjabi root word of each node to Universal Words and to retrieve its lexical semantic information. Exact UW is
extracted from the dictionary on the basis of node’s Punjabi root word attribute and its grammatical category (extracted from the parser). Since, Punjabi-UW dictionary may contain more than one entry for a given Punjabi word, the searching process retrieves the UW that matches with the node’s Punjabi word and its grammatical category. For example, Punjabi word, ਖੇਡ khēḍ ‘play’ has two entries in Punjabi-UW dictionary, one as a noun and other as a verb. It selects only that entry which matches with the grammatical category of the node given by the Punjabi shallow parser. If a node is marked as unknown in the first phase of parsing, then node’s Punjabi word attribute is searched in dictionary with its grammatical category as ‘null’. In case of multiple entries of that word, the system returns the UW of first entry and thus the unknown word becomes known during this phase. After extracting the UW, the node’s UW attribute is updated and linked list of lexical and semantic attributes is extended to append the UW dictionary attributes with the attributes generated by the parser.

After this phase, the linked list for the example sentence given in (4.3) is shown in (4.11).  
Node₁ Attributes: Punjabi root word: ਛੋਟਾ chōṭā; UW: little; POS: ADJ; Lexical and semantic attribute list: m, sg, o, ADJ.  
Node₂ Attributes: Punjabi root word: ਬੱਚਾ baccā; UW: child; POS: NN; Lexical and semantic attribute list: n, m, pl, 3, o, N, ANIMT.  
Node₃ Attributes: Punjabi root word: ਨੇ nē; UW: null; POS: PSP; Lexical and semantic attribute list: psp, d.  
Node₄ Attributes: Punjabi root word: ਕਿਤਾਬ kitāb; UW: book; POS: NN; Lexical and semantic attribute list: n, f, sg, 3, d, N, INANI.  
Node₅ Attributes: Punjabi root word: ਪਾਠ path; UW: study(icl>do); POS: VM; Lexical and semantic attribute list: v, f, sg, any, V.  
Node₆ Attributes: Punjabi root word: ਇਆ iā; UW: null; POS: null; Lexical and semantic attribute list: null. 

4.5.4 Case marker lookup phase

If Punjabi root word attribute of a node is not found in the Punjabi-UW dictionary, then it may be a case marker or function word of the language having no corresponding UW. In such a case, node’s Punjabi word attribute is searched in the case marker lookup file.
If a word is found then the information about the case marker is added in the linked list of lexical and semantic attributes of the node and its UW is set to ‘null’ (because a case marker has no corresponding UW). This information plays an important role in resolving UNL relations in UNL generation phase.

4.5.5 Unknown word handling phase

As discussed earlier, there might be some words in the input sentence, which would not have been resolved by Punjabi shallow parser. These words are marked as ‘unk’ (unknown) by the parser. If an unknown word is resolved in Universal Word lookup phase, then corresponding node is updated with its UW and dictionary attributes. If unknown words are not resolved in the Universal Word lookup phase, these are resolved in the Case marker lookup phase. If some words still remain unknown, then, these words are processed in Unknown word handling phase.

In this phase, system searches an unknown word in unknown word handling file. It contains only those Punjabi words that are derived from some root words because all other unknown words are resolved by UW lookup phase or Case marker lookup phase. The unknown word handling file stores Punjabi words with their corresponding root words, e.g., ਜਾਿਾਾਂਗਾ jāvāṅgā ‘will go’ is stored with ਜਾ ja ‘go’.

If an unknown word is a derived form of a root word, then its root word is retrieved from this lookup process. The system replaces the unknown word of node’s Punjabi word attribute with the root word extracted from the lookup process. The modifier of the root word is extracted by the system and stored as new node in the linked list.

For example, in case of unknown Punjabi word, ਜਾਿਾਾਂਗਾ jāvāṅgā ‘will go’ having root word ਜਾ ja ‘go’ has ਵਾਂਗਾ vāṅgā as a modifier. This modifier contains tense, number and gender information about the sentence. It plays an important role in the generation of UNL attributes (Ali et al., 2008). Thus, a new node is inserted in the linked list for this modifier as Punjabi root word attribute and its UW attribute, POS attribute and linked list of lexical semantic attributes are all set to ‘null’. As such, in case of unknown word ਜਾਿਾਾਂਗਾ jāvāṅgā ‘will go’, node’s Punjabi word attribute is set to ਜਾ ja ‘go’ and a new node is inserted into the linked list with its Punjabi word attribute as ਵਾਂਗਾ vāṅgā.
If a node is updated by unknown word handling phase, it is again processed in Universal Word lookup phase for getting its UW otherwise the token remains to be unknown as shown in Figure 4.2.

4.5.6 User interaction phase
In this optional phase, user is requested to supply information for unknown nodes. The system prompts all unknown nodes to user and requests for its UW and lexical semantic attributes’ information. If user supplies required information, the system stores it in the node’s data structure and starts the UNL generation phase; otherwise system tries to generate the UNL expression with the unknown nodes.

4.5.7 UNL generation phase
After creation of linked list and its processing in above discussed phases, the linked list is ready for the generation of UNL expression. This phase uses nine hundred EnConverter analysis rules for generation of UNL expression. Some of these rules are given in Appendix-A. This phase uses algorithm 4.1 for UNL relation resolution and generation of attributes.

Algorithm 4.1: UNL relation resolution and generation of attributes

i) Process each node of linked list by considering the first node as left analysis window and the node next to this as right analysis window.

ii) Search for the required rule from the set of EnConverter analysis rules depending upon the conditions of left and right analysis windows.

iii) Fire the EnConverter analysis rule if the conditions of left and right analysis windows match with lexical semantic attributes of the corresponding nodes of the linked list. If no rule is fired, then go to step (vi).

iv) Perform the actions specified in the fired rule to modify the linked list to resolve the UNL relations and generate UNL attributes.

v) Consider first node of modified linked list as left analysis window and node next to this as right analysis window. Go to step (ii) with new analysis windows. If the modified linked list contains a single node only, then consider that node as entry node and stop further processing. It means that all the nodes are successfully processed by the system.
vi) If no rule is fired in step (ii), then shift the window to right. This effectively means that right analysis node will become the left analysis window and node next to this will become the right analysis window. Go to step (ii) with new analysis windows.

Generation of UNL expression from input Punjabi sentence has been explained in the next section with the help of one example sentence.

4.6 EnConversion of a simple Punjabi sentence to UNL expression
The process of EnConversion of input Punjabi sentence to UNL expression is illustrated with an example sentence given in (4.12).

Example Punjabi sentence: ਮੈਂ ਗੈਰੇਜ ਵਿਚ ਕਾਰ ਧੋਂਦਾ ਹਾਂ। …(4.12)

Transliterated Punjabi sentence: maiṃ gairēj vic kār dhōndā hāṃ.
Equivalent English sentence: I wash the car in garage.
Processing of this sentence in various phases of Punjabi EnConverter is explained below.

i) Parser phase
The input sentence is processed by Punjabi shallow parser. The output of pruning stage of parser is given in (4.13).

Final Output

```
1  ((  NP
1.1 ਮੈਂ  PRP  <fs af='ਮੈਂ,pn,any,sg,1,d,,'>
     ))
2  ((  NP
2.1 ਗੈਰੇਜ  NN  <fs af='ਗੈਰੇਜ,unk,,,,,' poslcat="NM"> 
2.2 ਵਿਚ  PSP  <fs af='ਵਿਚ,psp,any,sg,,d,,'>
     ))
3  ((  NP
3.1 ਕਾਰ  NN  <fs af='ਕਾਰ,n,f,sg,3,d,,'>
     ))
4  ((  VGF
4.1 ਧੋਂਦਾ  VM  <fs af='ਧੋ,v,m,sg,any,,ਦਾ,ਦਾ'> 
4.2 ਹਾਂ  VAUX  <fs af='ਹਾਂ,n,f,sg,3,d,,' poslcat="NM"> 
```
ii) Linked list creation phase

In this phase, tokens ਮੈਂ 'i', ਗੈਰੇਜ 'garage', ਵਿਚ 'in', ਕਾਰ 'car', ਦਾ 'wash', ਹਾਂ act as the candidates for the nodes of linked list. The set of largest consecutive words are searched in Punjabi-UW dictionary and verb-modifier database. A linked list with six nodes is created in this phase. The details of this linked list are given in (4.14).

Node\_1 Attributes: Punjabi root word: ਮੈਂ; UW: null; POS: PRP; Lexical and semantic attribute list: pn, any, sg, 1, d.

Node\_2 Attributes: Punjabi root word: ਗੈਰੇਜ; UW: null; POS: NN; Lexical and semantic attribute list: unk.

Node\_3 Attributes: Punjabi root word: ਵਿਚ; UW: null; POS: PSP; Lexical and semantic attribute list: psp, any, sg, d.

Node\_4 Attributes: Punjabi root word: ਕਾਰ; UW: null; POS: NN; Lexical and semantic attribute list: n, f, sg, 3, d.

Node\_5 Attributes: Punjabi root word: ਦਾ ਹਾਂ; UW: null; POS: VM; Lexical and semantic attribute list: v, m, sg, any.

Node\_6 Attributes: Punjabi root word: ਦਾ ਹਾਂ; UW: null; POS: null; Lexical and semantic attribute list: n, f, sg, 3, d.

It has been noted that Node\_2 corresponding to Punjabi word ਗੈਰੇਜ 'garage' is classified into an unknown node, since the word is not resolved by Punjabi shallow parser. Node\_6 contains two Punjabi words ਦਾ and ਹਾਂ, because these two consecutive root words are found as single entry in verb-modifier database, so a single node Node\_6 has been created for this group.

iii) Universal Word lookup phase

In this phase, Universal Word (UW) for each node’s Punjabi root word attribute is searched from the Punjabi-UW dictionary. The POS category of the node extracted from the parser is also used during this search. The exact match is found to extract UW for the
specified POS category of the word from Punjabi-UW dictionary. The extracted UW is loaded into the node’s UW attribute and node’s lexical semantic attribute list is also extended with attributes extracted from the Punjabi-UW dictionary. After processing all the nodes in the Universal Word lookup phase, the linked list is shown in (4.15).

Node 1 Attributes: Punjabi root word: ਮੈਂ mai; UW: I(iclperson)); POS:  PRP; Lexical and semantic attribute list: pn, any, sg, 1, d, PERPRON, ANIMT.

Node 2 Attributes: Punjabi root word: ਗਾਰੇਜ gairēj; UW: garage(icl>thing)); POS:  NN; Lexical and semantic attribute list: unk, N, ANIMT, PLC.

Node 3 Attributes: Punjabi root word: ਵਿਚ vic; UW: null; POS:  PSP; Lexical and semantic attribute list: psp, any, sg, d.

Node 4 Attributes: Punjabi root word: ਕਾਰ kār; UW: car(icl>thing)); POS:  NN; Lexical and semantic attribute list: n, f, sg, 3, d, N.

Node 5 Attributes: Punjabi root word: ਧੋ dhō; UW: wash(icl>do); POS:  VM; Lexical and semantic attribute list: v, m, sg, any,V.

Node 6 Attributes: Punjabi root word: ਦਾ ਹਾਾਂ dā hā; UW: null; POS:  null; Lexical and semantic attribute list t: n, f, sg, 3, d. …(4.15)

The unknown word ਗਾਰੇਜ gairēj ‘garage’ in Node 2 becomes known in this phase as ਗਾਰੇਜ gairēj ‘garage’ has an entry in Punjabi-UW dictionary.

iv) Case marker lookup phase
Those nodes which are not found in the Punjabi-UW dictionary lookup phase are processed in this phase. It means that Node 3 and Node 6 are candidates for processing in this phase. These nodes do not have any entry in case marker lookup file, and are also not categorized as unknown words by the parser. As such, the information given by the parser is sufficient for processing.

v) Unknown word handling phase
At this stage, there is no unknown node in the linked list. Thus, unknown word handling phase and user interaction phase are not invoked for this example sentence.

vi) UNL generation phase
In this phase algorithm 4.1, given in Section 4.5.7, is used for the processing of linked list to resolve UNL relation and to generate UNL attributes.
The intermediate steps for generation of UNL expression, for example sentence given in (4.12), are given below. Here, the node-list is shown within ‘<<’ and ‘>>’ and the analysis windows are denoted within ‘[’ and ‘]’. Initially, the node-list will have the structure given in (4.16).

<<[मैं] [कैगेड] विग नां र्नद>> …(4.16)

<<[mai] [gairēジー] vic kār dhō dā hām>>

Here, no EnConversion rule is fired between left and right analysis windows. Now, the analysis windows are shifted to right. Thus, the node-list will become as shown in (4.17).

<<मैं [कैगेड] विग नां र्नद>> …(4.17)

<<mai [gairēジー] [vic] kār dhō dā hām>>

Now, left composition rule given in (4.18) is fired between left and right analysis windows.

+{N,PLC:+WICH:null}{{विग:null:.null}} …(4.18)

This rule is preceded by ‘+’ sign, which indicates that it is a left composition rule. It results into concatenation of right node to the left node as a single composition node and the attributes of left node are inherited for further processing. The condition field of right analysis window contains a Punjabi word in ‘[’ and ‘]’, means that the rule will fire if the right analysis window has Punjabi root word attribute value विग vic. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attribute in lexical semantic attribute list. Now, ‘WICH’ (indicating Punjabi case marker) is added as attribute of left analysis window and the node-list will become as shown in (4.19).

<<[मैं] [कैगेड_विग] विग नां र्नद>> …(4.19)

<<[mai] [gairēジー_vic] kār dhō dā hām>>

Here, no EnConversion rule is fired between left and right analysis windows, and the analysis windows are shifted to right. The node-list after shifting to right is given in (4.20).

<<मैं [कैगेड_विग] विग नां र्नद>> …(4.20)

<<mai [gairēジー_vic] [kār] dhō dā hām>>

Again, no EnConversion rule is fired between left and right analysis windows, and the
analysis windows are shifted to right. The node-list after shifting to right is given in (4.21).

\[
\text{\textless \textless मैं गैरेज_विच [चर] [चे] धो दा} \text{\textgreater \textgreater} \quad \ldots(4.21)
\]

\[
\text{\textless \textless मैं gairēj_vic [kār] [dhō] dā hā} \text{\textgreater \textgreater}
\]

Now, the analysis windows trigger the right modification rule given in (4.22).

\[
>\{N,\text{INANI,}^\text{WICH,}^\text{PLC,}^\text{SRCRES,}^\text{TOON}\text{null:obj}\} \{V:+OBJRES: \text{null}\} \quad \ldots(4.22)
\]

As this rule is preceded by ‘>’, it is a right modification rule. This rule is applicable when left node modifies the right node. It deletes the left node from the node-list, while the right node remains in the node-list. The presence of ‘^’ before the dictionary attributes in the condition part of rule indicates that these attributes should not be present in corresponding node’s lexical semantic attribute list. Here, ‘obj’ relation is resolved between two analysis windows (due to presence of ‘obj’ in the relation part of left analysis window) as shown in (4.23).

\[
\text{obj(wash(icl>do), car(icl>thing))} \quad \ldots(4.23)
\]

The presence of ‘+’ sign in the action part of right analysis window results into the addition of ‘OBJRES’ (indicating node is involved in ‘obj’ relation) attribute to right analysis window and the node-list becomes as shown in (4.24).

\[
\text{\textless \textless मैं [कैंटेन_विच] घे सभ दा} \text{\textgreater \textgreater} \quad \ldots(4.24)
\]

\[
\text{\textless \textless मैं gairēj_vic [धो] dā hā} \text{\textgreater \textgreater}
\]

At this stage, no EnConversion rule is fired between left and right analysis windows, and the analysis windows are shifted to right. The node-list after shifting to right is given in (4.25).

\[
\text{\textless \textless मैं [कैंटेन_विच] [चे] धो दा} \text{\textgreater \textgreater} \quad \ldots(4.25)
\]

\[
\text{\textless \textless मैं gairēj_vic [dhō] dā hā} \text{\textgreater \textgreater}
\]

The rule given in (4.26) is fired between left and right analysis windows at this stage.

\[
>\{N,\text{PLC,}^\text{WICH:null:plc}\} \{V:\text{null:null}\} \quad \ldots(4.26)
\]

It is a right modification rule that deletes the left node from the node-list, while the right node remains in the node-list. Here, ‘plc’ relation as given in (4.27) is resolved between two analysis windows.

\[
\text{plc(wash(icl>do),garage(icl>thing))} \quad \ldots(4.27)
\]
The node-list now becomes as shown in (4.28).

\[
\text{[[में] [धो] दाहां]}} \quad \text{\ldots (4.28)}
\]

\[
\text{[[में] [धो] दाहां]} \quad \text{\ldots (4.28)}
\]

Now, the rule given in (4.29) is fired between left and right analysis windows.

\[
> \{\text{PERPRON,ANIMT:null:agt}\} \{\text{V:+AGTRES:null}\} \quad \text{\ldots (4.29)}
\]

It is again a right modification rule. It deletes the left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation as given in (4.30) is resolved between two analysis windows.

\[
\text{agt(wash(icl>do), I(icl<person))} \quad \text{\ldots (4.30)}
\]

The node-list now becomes as shown in (4.31).

\[
\text{[[धो] [दाहां]]}} \quad \text{\ldots (4.31)}
\]

\[
\text{[[धो] [दाहां]} \quad \text{\ldots (4.31)}
\]

Now, the left composition rule given in (4.32) is fired between left and right analysis windows.

\[
+ \{\text{V:+.@present.@custom.@sg.@male:null}\} \{\text{चन जन:null:null}\} \quad \text{\ldots (4.32)}
\]

It is also a left composition rule which results into concatenation of right node to the left node as a single composition node and the attributes of left node are inherited for further processing. The condition field of right analysis contains a Punjabi word ‘चन जन’ ‘दाहां’ in ‘[’ and ‘]’, it means that the rule will fire if the right analysis window has Punjabi root word attribute ‘चन जन’ ‘दाहां’. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these attributes are preceded by ‘.@’ sign, they are concatenated to corresponding UW as UNL attributes. Now, the UW ‘wash(icl>do)’ is modified as ‘wash(icl>do).@present.@custom.@sg.@male’ and the node-list becomes as shown in (4.33).

\[
\text{[[चन जन]]}} \quad \text{\ldots (4.33)}
\]

\[
\text{[[चन जन]} \quad \text{\ldots (4.33)}
\]

Now, there is a single node in the node-list. This is considered as root node and ‘.@entry’ attribute is concatenated to its UW as given in (4.34).

\[
\text{wash(icl>do).@present.@custom.@sg.@male.@entry} \quad \text{\ldots (4.34)}
\]

\[
\text{wash(icl>do).@present.@custom.@sg.@male.@entry} \quad \text{\ldots (4.34)}
\]
Finally, the UNL expression is generated by the Punjabi EnConverter system for input Punjabi sentence as given in (4.35).

Punjabi sentence: ਮੈਂ ਗਾਰੇਜ ਵਿਚ ਕਾਰ ਧੋਂਦਾ ਹਾਂ।
Transliterated sentence: mai[m gairēj vic kār dhōndā hām.
Equivalent English sentence: I wash the car in garage.

UNL expression generated by the system:

{unl}

obj(wash(icl>do).@sg.@male.@present.@entry, car(icl>thing))
plc(wash(icl>do).@sg.@male.@present.@entry, garage(icl>thing))
agt(wash(icl>do).@sg.@male.@present.@entry, I(icl<person))

{/unl}  ...(4.35)

The UNL graph of the UNL expression given in (4.35) is shown in Figure 4.4.

![UNL graph for the UNL expression in (4.35)](image)

**Figure 4.4: UNL graph for the UNL expression in (4.35)**

### 4.7 EnConversion of complex sentences

A complex sentence is the sentence that has one main clause and one or more subordinate clauses. An important issue in the analysis of complex sentences is to find the words that act as the clause delimiters and to relate the sub-clause to the main clause of the sentence using appropriate UNL relation, e.g., an adverb clause should be related with the verb of main clause with ‘plc’ or ‘tim’ or ‘man’ or ‘con’ UNL relation and an adjective clause should be related with the noun of main clause with ‘aoj’ relation.

Three basic types of subordinate (or dependent) clauses, namely, noun clause, adjective clause and adverb clause have been considered in this work. In this section, the EnConversion process for each of these clauses is presented.
4.7.1 EnConversion process of noun clause sentence

A noun clause sentence is identified when the object of the main verb is a complete meaningful sentence in itself. The noun clause is resolved between the verb of the main clause and the subordinate clause. Punjabi EnConverter uses the attribute ‘NCL’ (noun clause) for identifying noun clause delimiter. Punjabi has a word ਵਕ ਦਾ ‘that’ to represent a noun clause delimiter. This word indicates the beginning of subordinate clause and shall have the attribute ‘NCL’.

The process of EnConversion of noun clause sentences is illustrated with an example Punjabi sentence given in (4.36).

Example Punjabi Sentence: ਰਾਮ ਨੇ ਵਕਹਾ ਵਕ ਸੁਖਵਿੰਦਰ ਚੂੰਗਾ ਹੈ। …(4.36)

Transliterated Punjabi sentence: rām nē kihā ki sukkhvindar caṅgā hai.

Equivalent English Sentence: Ram said that Sukhwinder is good.

Punjabi shallow parser is used to parse the example sentence and a linked list of seven nodes, i.e., ਰਾਮ rām ‘Ram’, ਨੇ nē, ਵਕਾਁ rē kihā ‘said’, ਵਕਾਂ ਵਕ ਸੁਖਵਿੰਦਰ ਚੂੰਗਾ caṅgā ‘good’ and ਹੈ hai ‘is’ are created for generation process. The intermediate steps for the generation of UNL expression for this example sentence is shown in (4.37). Here, node-list is enclosed within ‘<<’ and ‘>>’; the analysis windows are enclosed within ‘[‘ and ‘]’.

<<[ਰਾਮ_ਨੇ] ਕਵਹ ਇਆ ਵਕ ਸੁਖਵਿੰਦਰ ਚੂੰਗਾ ਹੈ>> …(4.37)

<<[rām_nē] kahi iā ki sukkhvindar caṅgā hai>>

At the outset, left composition rule given in (4.38) is fired between left and right analysis windows.

++{N, ANIMT:+CASE:null}{CASE:+null:null} …(4.38)

It results into concatenation of right node to the left node as a single node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into addition of attribute into the lexical semantic attribute list of node at left analysis window. Here, ‘CASE’ is added as an attribute to left analysis window node and the node-list will become as shown in (4.39).

<<[ਰਾਮ_ਨੇ] ਕਵਹ ਇਆ ਵਕ ਸੁਖਵਿੰਦਰ ਚੂੰਗਾ ਹੈ>> …(4.39)

<<[rām_nē] kahi iā ki sukkhvindar caṅgā hai>>
Now, the rule given in (4.40) is fired between left and right analysis windows.

\[ \{N,ANIMT,CASE,^CAG,^PSP,^POS,^KI,^TCLB,^TCL,^CCLB,^CCL:\text{null};agt\} \]
\[ \{V,^AGTRES,^VOCURR,^COO:+AGTRES:\text{null}\} \] …(4.40)

It is a right modification rule resulting into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation as given in (4.41) is resolved between two analysis windows.

agt(say, Ram(icl>person)) …(4.41)

The node-list after processing with right modification rule will become as shown in (4.42).

<<[चति रिशा] [ति] मूँहीरिस्त चंगा श्री>> …(4.42)

<<[kahi iā] [ki] sukkhvindar caṅgā hai>>

At this stage, no EnConversion rule is fired between left and right analysis windows, and the analysis windows are shifted to right. The node-list after shifting to right is given in (4.43).

<<[चति रिशा] [ति] मूँहीरिस्त चंगा श्री>> …(4.43)

<<kahi iā [ki] sukkhvindar caṅgā hai>>

At this point, system fires the right composition rule given in (4.44) to add ‘NCL’ to the right analysis window node.

-[[ति]:null:null} {N,ANIMT:+NCL:null} …(4.44)

This rule results into concatenation of left node to the right node as a single composite node and the attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. After the application of this rule, noun will get the attributes ‘NCL’ in addition of its existing attributes and the node-list will become as shown in (4.45).

<<[चति रिशा] [ति_मूँहीरिस्त] चंगा श्री>> …(4.45)

<<[kahi iā] [ki_sukkhvindar] caṅgā hai>>

At this stage, no EnConversion rule is fired between left and right analysis windows, and the analysis windows are shifted to right. The node-list after shifting to right is given in (4.46).

106
Now, the rule given in (4.47) is fired between left and right analysis windows.

\[
\{N,^\text{TIME},^\text{TOON},^\text{WICH},\text{NCL:}null:\text{aoj}\} \{\text{ADJ},\text{ST:}+\text{AOJRES},^\text{NCL:}null\} \quad \text{…(4.47)}
\]

The right modification rule deletes left node from the node-list, while the right node remains in the node-list. Here, ‘aoj’ relation as given in (4.48) is resolved between two analysis windows.

\[
\text{aoj}(\text{good}(\text{icl>state}), \text{Sukhwinder}(\text{icl>person})) \quad \text{…(4.48)}
\]

The node-list after processing with right modification rule will become as shown in (4.49).

<<[ਕਵਹ ਇਆ] [ਚੂੰਗਾ] ਹੈ>> \quad \text{…(4.46)}

<<[kahi iā] [caṅgā] hai>>

Now, the right modification rule given in (4.50) is fired between left and right analysis windows to resolve the relation ‘obj’ between two UWs as shown in (4.51).

\[
\{V:+.\text{@entry:}null\} \{\text{ADJ},\text{NCL:}+\text{COMPLEX:}obj\} \quad \text{…(4.50)}
\]

\[
\text{obj}(\text{say.}\text{@entry}, \text{good}) \quad \text{…(4.51)}
\]

The rule given in (4.50) resolves a UNL relation ‘obj’ between main verb and subordinate noun clause. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, this attribute is preceded by ‘.@’ sign, it is concatenated to corresponding UW as UNL attribute. Here, the UW ‘say’ is modified as ‘say.@entry’. At this stage, due to presence of main clause and sub-ordinate clause in a UNL relation, system resolves a composite UW ‘:01’ for UW ‘good’ and it is replaced with ‘:01’ as shown in (4.52) and system converts earlier resolved relation ‘aoj’ involving UW ‘good’ as composite relation as shown in (4.53).

\[
\text{obj}(\text{say.}\text{@entry}, :01) \quad \text{…(4.52)}
\]

\[
\text{aoj}:01(\text{good}, \text{Sukhwinder}(\text{icl>person})) \quad \text{…(4.53)}
\]

After the application of rule (4.50), the node-list will become as shown in (4.54).

<<[ਚੂੰਗਾ] [ਚੇ]>> \quad \text{…(4.54)}

<<[caṅgā] [hai]>>
Here, left composition rule given in (4.55) is fired between left and right analysis windows.

\[ +\{\text{ADJ:+VAUX,+.@present.@sg:null}\} \{[ਚੂੰਗਾ:null:.null]\} \]

\[ \text{…(4.55)} \]

It results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign beginning with ‘.@’ in the action part of left analysis window results into concatenation of ‘.@present.@sg’ to the corresponding UW, i.e., the UW ‘good’ will become ‘good.@present.@sg’. The node-list after application of rule given in (4.55) will become as shown in (4.56).

\[ <<[ਚੂੰਗਾ_ਹੈ]>><\text{…(4.56)}\]

\[ <<[caṅgā_ hai]>><\text{…(4.56)}\]

Since, it is the last node so ‘.@entry’ will be added to UW ‘good.@present.@sg’ and UNL generation process is completed at this stage, resulting into an equivalent UNL expression as given in (4.57).

\{unl\}

agt(say.@entry, Ram(icl>person))
aoj:01(good(icl>state).@present.@sg.@entry, Sukhwinder(icl>person))
obj(say.@entry, :01)
\{/unl\} \text{…(4.57)}

Here, the scope node ‘:01’ acts as the object of main sentence and it is used to represent ‘Sukhwinder is good’, a complete meaningful sentence. The UNL graph for UNL expression (4.57) is shown in Figure 4.5.

![UNL graph for UNL expression given in (4.57)](image-url)
4.7.2 EnConversion process of adverb clause sentences

Adverb clause is a sub-ordinate clause that acts as adverb in a sentence. It may modify some verb, adjective or adverb in the main clause. In order to resolve UNL relations for adverb clauses, these clauses are classified as adverb clause for time, adverb clause for condition, adverb clause for manner and adverb clause for place (Giri, 2000). The types of adverb clauses, along with attributes used and corresponding Punjabi word delimiters to represent these types, are depicted in Table 4.1.

<table>
<thead>
<tr>
<th>Type of clause</th>
<th>Attributes used</th>
<th>Punjabi words used to represent adverb clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Adverb Clause</td>
<td>TCLB, TCL</td>
<td>ਸ਼੍ਰੇ jadōṃ ‘when’ and ਓਦੋਂ Ṽdōṃ ‘then’</td>
</tr>
<tr>
<td>Conditional Adverb Clause</td>
<td>CCLB, CCL</td>
<td>ਨਾਵੇਖਾਸ jēkar/jē ‘if’ and ਟਾੰਮ ਥੇ ‘then’</td>
</tr>
<tr>
<td>Manner Adverb Clause</td>
<td>MCLB, MCL</td>
<td>ਸ਼੍ਰੇ jiddāṃ and ਓਦੋਂ Ṽdōṃ</td>
</tr>
<tr>
<td>Place Adverb Clause</td>
<td>PCLB, PCL</td>
<td>ਸ਼੍ਰੇ jithē and ਓਦੋਂ utthē</td>
</tr>
</tbody>
</table>

The EnConversion process of each of these types of adverb clauses is described in subsequent sub-sections.

4.7.2.1 EnConversion process of sentences containing adverb clause for time

The adverb clause for time is resolved between the verb of main clause and the verb of sub-ordinate clause using the attribute ‘TCL’ (time clause). The words ਸ਼੍ਰੇ jadōṃ ‘when’ and ਓਦੋਂ Ṽdōṃ ‘then’ act as the clause delimiter for adverb clause for time in Punjabi.

The word, ਸ਼੍ਰੇ jadōṃ ‘when’, indicates the beginning of time condition and is represented by ‘TCLB’ attribute, while word, ਓਦੋਂ Ṽdōṃ ‘then’, indicates the action as a result of that time and is represented by ‘TCL’ attribute.

The handling of adverb clause of time is illustrated with an example Punjabi sentence given in (4.58).

Example Punjabi sentence: ਸ਼ਹੇ ਨੇ ਇਜਾਨ ਨੇ ਭਾਜਾਂ ਨਾਨਕ ਨੁਹ ਬਾਜਾਰ ਨੂੰ ਜਾਏਂਗਾ।           …(4.58)
Transliterated Punjabi sentence: *jadōṃ maiṃ kālaj tōṃ āvāṅgā ńdōṃ tūṃ bāzār nūṃ jāvēṅgā.*

Equivalent English sentence: When I will come from college then you will go to market.

The input sentence will be processed with parser phase, linked list creation phase, UW lookup phase, case marker lookup phase and unknown word handling phase of Punjabi EnConverter. After this, the node-net will be ready for UNL generation phase. The process of UNL generation phase for example sentence indicating node-net, rule fired and action taken by the fired rule at each iteration is depicted in Table 4.2. Here, left and right analysis windows are enclosed within ‘[’ and ‘]’.

Table 4.2: EnConversion process of adverb clause for time for example sentence given in (4.58)

<table>
<thead>
<tr>
<th>Iteration:1</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ਜਦੋਂ] [ਮੈਂ] ਕਾਲਜ ਤੋਂ ਆ ਅੇਗਾ ਚੀਂ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾਏਗਾ</td>
<td>{-TCLB,^V,^N,^PRON,^ADJ,^ADV:null:null} {PERPRON:+TCLB:null}</td>
<td>This rule is preceded by ‘-’ sign, which indicates that it is a right composition rule. It results into concatenation of left node to the right node as a single composite node and the attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. This rule concatenates the adverb of time delimiter, recognized by the presence of the attribute ‘TCLB’ to the personal pronoun on the right. After the application of this rule, personal pronoun gets the attributes ‘TCLB’ in addition to its existing attributes. When other relations are resolved with this personal pronoun, it will retain this attribute for further processing.</td>
</tr>
<tr>
<td></td>
<td>[ਜਦੋਂ] [ਮੈਂ] ਕਾਲਜ ਤੋਂ ਆ ਅੇਗਾ ਚੀਂ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾਏਗਾ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:2</th>
<th>Node-net</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ਜਦੋਂ_ਮੈਂ][ਕਾਲਜ] ਤੋਂ ਆੇਗਾ ਚੀਂ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾਏਗਾ</td>
<td></td>
</tr>
<tr>
<td>Iteration: 3</td>
<td>Node-net</td>
<td>ज्ञांम_में [कलाज] तोमा एगा ओडोम तुम्ह बाज़र नुम जा एगा</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Rule fired</td>
<td>No rule fired.</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>The analysis windows are shifted to right.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 4</th>
<th>Node-net</th>
<th>ज्ञां_में [कलाज] तोमा एगा ओडोम तुम्ह बाज़र नुम जा एगा</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>+{N:+TOON:null} {{null:null}}</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>It is a left composition rule and results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of ‘TOON’ attribute in lexical semantic attribute list of node at left analysis window.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 5</th>
<th>Node-net</th>
<th>ज्ञां_में [कलाज] तोमा एगा ओडोम तुम्ह बाज़र नुम जा एगा</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>&gt;{N,TOON,PLC,^FMTRES,^TIME,^DUR:null:plf} {V:+PLFRES:null}</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, 'plf' relation is resolved between</td>
<td></td>
</tr>
</tbody>
</table>
two analysis windows as shown below.
plf(come(icl\(\rightarrow\)do), college(icl\(\rightarrow\)school))

<table>
<thead>
<tr>
<th>Iteration: 6</th>
<th>Node-net</th>
</tr>
</thead>
</table>
|             | [ਜਦੋਂ ਮੈਂ] [ਆ ਏਗਾ ਓਦੋਂ ਤੋਂ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ ] 
| Rule fired   | >{PERPRON,ANIMT,TCLB,\(^{\text{CAG}}\),\(^{\text{NOO}}\):null:agt} 
| Action taken | It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows as shown below. 
|             | 
| Iteration: 7 | Node-net | +{V,m,sg+:\text{@future.@sg.@male}:null} 
| Rule fired   | No rule fired. 
| Action taken | The analysis windows are shifted to right. 
| Iteration: 9 | Node-net | 
|             | [ਆ_ਏਗਾ] [ਓਦੋਂ] 

|          | [ ਅ ] [ ਏਗਾ ] ਓਦੋਂ ਤੋਂ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ ] 
| Rule fired | No rule fired. 
| Action taken | The analysis windows are shifted to right. 
| Iteration: 9 | Node-net | 
|             | [ਆ_ਏਗਾ] [ਓਦੋਂ] 

\(^{\text{CAG}}\) = \text{Case}
\(^{\text{NOO}}\) = \text{Nominal Object}
\(^{\text{TCLB}}\) = \text{Technical Language Block}
ā_ ēgā [ ōdōm] [tūm] bāzār nūm jā ēgā

Rule fired

No rule fired.

Action taken

This right composition rule results into concatenation of left node to the right node as a single composite node and the attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. This rule concatenates the adverb of time delimiter, recognized by the presence of the attribute ‘TCL’ to the pronoun on the right. After the application of this rule, pronoun gets the attributes ‘TCL’ in addition of its existing attributes. When other relations are resolved with this pronoun, it will retain this attribute for further processing.

Iteration:10

Node-net

ā_ ēgā [ ōdōm_tūm] bāzār nūm jā ēgā

Rule fired

No rule fired.

Action taken

The analysis windows are shifted to right.

Iteration:11

Node-net

ā_ ēgā [ ōdōm_tūm] [bāzār] nūm jā ēgā

Rule fired

No rule fired.

Action taken

The analysis windows are shifted to right.

Iteration:12

Node-net

ā_ ēgā ōdōm_tūm [bāzār] [nūm] jā ēgā

Rule fired

+{N:+NOO:null}{[null]}:null :null

Action taken

It is a left composition rule and results into concatenation of right node to the left node as a single composite node
and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of ‘NOO’ attribute in lexical semantic attribute list of node at left analysis window.

<table>
<thead>
<tr>
<th>Iteration: 13</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ā_ ēgā] [ōdōm_tūm] bāzār nūṃ jā ēgā</td>
<td>No rule fired.</td>
<td>The analysis windows are shifted to right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 14</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ā_ ēgā [ōdōm_tūm] [bāzār nūṃ] jā ēgā</td>
<td>No rule fired.</td>
<td>The analysis windows are shifted to right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 15</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ā_ ēgā ōdōm_tūm [bāzār nūṃ] jā ēgā</td>
<td>&gt; {N,PLC:null:plt} {V,jA:+PLTRES:null}</td>
<td>It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘plt’ relation is resolved between two analysis windows as shown below. plt(go(icl&gt;do), market)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 16</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ā_ ēgā [ōdōm_tūm] jā ēgā</td>
<td>No rule fired.</td>
<td>The analysis windows are shifted to right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 17</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ā_ ēgā [ōdōm_tūm] jā ēgā</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rule fired</td>
<td>Rule fired</td>
<td>Action taken</td>
<td>Action taken</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>ā_ēgā [ōdōm_tūm] [jā] ēgā</td>
<td>&gt;{PRON,ANIMT,TCL,^CAG,^NOO,^PSP,sg.^CCLB,&lt;^CCL:null:agt} {V,^VAUX,^AGTRES,^VOCURR,^COO:+AGTRES,&lt;+TCL:null}</td>
<td>It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows as shown below. agt(go(icl&gt;do), you(icl&gt;person))</td>
<td>This right modification rule deletes the left node from the node-list and resolves a relation between left analysis having a main verb with adverb clause beginner time delimiter ‘TCLB’ attribute and right analysis window having sub-ordinate verb with adverb clause ending time delimiter attribute ‘TCL’. Thus, a ‘tim’ relation is resolved between main verb and sub-ordinate adverb clause of time. Due to presence of main clause and sub-ordinate clause in a UNL relation, a composite UW ‘:01’ is resolved by the system for UW ‘go’ and it replaces UW ‘go’ with ‘:01’ as shown below in resolved relation ‘tim’. Here, the system also converts earlier resolved UNL relations involving ‘go’ as composite relation as shown below. tim(come(agt&gt;thing), :01) plt:01(go(icl&gt;do), market) agt:01(go(icl&gt;do), you(icl&gt;person))</td>
</tr>
<tr>
<td>Iteration:18</td>
<td>Node-net</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration: 19</td>
<td>Node-net</td>
<td>Rule fired</td>
<td>Action taken</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
|              | [ਜਾ] [ਏਗਾ] | +{V,m.sg:+.@future.@sg.@male:null}{{[ਏਗਾ]:null:null}} | This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these attributes are preceded by ‘.@’ sign, they are concatenated to corresponding UW as UNL attributes. Here, the UW ‘go(icl>do)’ is modified as ‘go(icl>do).@future.@sg.@male’.
|              | [ਜਾ_ਏਗਾ] |             |              |

<table>
<thead>
<tr>
<th>Iteration: 20</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ਜਾ_ਏਗਾ]</td>
<td>No rule fired.</td>
<td>Since, it is the last node so ‘.@entry’ will be added to UW to ‘go(icl&gt;do).@future.@sg.@male’ and the UNL generation process completes at this iteration.</td>
</tr>
</tbody>
</table>

After processing of input example sentence given in (4.58), as shown in Table 4.2, the system generates the equivalent UNL expression as given in (4.59).

```
{unl}
plf(come(agt>thing).@future.@sg.@male.@entry, college(icl>school))
agt(come(agt>thing).@future.@sg.@male.@entry, I(icl>person))
plt:01(go(icl>do).@future.@sg.@male.@entry, market)
agt:01(go(icl>do).@future.@sg.@male.@entry, you(icl>person))
tim(come(agt>thing).@future.@sg.@male.@entry, :01)
{/unl}  …(4.59)
```

Here, ‘:01’ indicates that it is a complex node. The UNL graph for UNL expression given in (4.59) is shown in Figure 4.6.
4.7.2.2 EnConversion process of sentences containing adverb clause for condition

The adverb clause for condition is resolved between the verb of main clause and the verb of sub-ordinate clause using the attribute ‘CCL’ (condition clause). The words ਜੇਕਰ/ਜੇ jēkar/jē ‘if’ and ਤਾਂ/ਤੇ tāṃ/tē ‘then’ act as the clause delimiters for adverb clause for condition in Punjabi. The word ਜੇਕਰ/ਜੇ jēkar/jē ‘if’ indicates the beginning of conditional clause and is represented by ‘CCLB’ attribute, while word ਤਾਂ/ਤੇ tāṃ/tē ‘then’ indicates the action as a result of that condition and is represented by ‘CCL’ attribute.

The handling of adverb clause for condition is illustrated with an example Punjabi sentence given in (4.60).

Example Punjabi sentence: ਜੇਕਰ ਤੂੰ ਮੇਰਾ ਸੁਜ਼ਾ ਨਹੀਂ ਮੂੰਵਨਾ ਤਾਂ ਤੂੰ ਪਛਤਾਿੇਂਗਾ। ... (4.60)

Transliterated Punjabi sentence: jēkar tūṃ mērā sujhāa nahī manniā tūṃ tūṃ pachtāvēṅgā.

Equivalent English sentence: If you will not agree to my suggestion then you will repent.

The input sentence will be processed with parser phase, linked list creation phase, UW lookup phase, case marker lookup phase and unknown word handling phase of Punjabi EnConverter. After this, the node-net will be ready for UNL generation phase. The process of UNL generation phase for example sentence indicating node-net, rule fired and action taken by the fired rule at each iteration is depicted in Table 4.3. Here, left and right analysis windows are again enclosed with in ‘[’ and ‘]’.
Table 4.3: EnConversion process of adverb clause of condition sentence for example sentence given in (4.60)

<table>
<thead>
<tr>
<th>Iteration:1</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[जेकर] [तू] मे मुझफ्फ़र रही मरह रिश्वा उं टुं पट्टा रेखा</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(jēkar) [tū] mai sujhāa nahī mann ìā tām tūmpachtā ēgā</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{{CCLB,^V,^N,^PRON,^ADJ,^ADV:null:null}}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{PRON:+CCLB:null}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This rule is preceded by ‘-’ sign, which indicates that it is a right composition rule. It results into concatenation of left node to the right node as a single composite node and the attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. This rule concatenates the adverb of condition delimiter, recognized by the presence of the attribute ‘CCLB’ to the pronoun on the right. After the application of this rule, pronoun gets the attributes ‘CCLB’ in addition of its existing attributes. When other relations are resolved with this pronoun, it will retain this attribute for further processing.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:2</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[जेकर_टू] [मि] मुझफ्फ़र रही मरह रिश्वा उं टुं पट्टा रेखा</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(jēkar_tū) [mai] sujhāa nahī mann ìā tām tūmpachtā ēgā</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No rule fired.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The analysis windows are shifted to right.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:3</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>नेकर_टू [मि] मुझफ्फ़र रही मरह रिश्वा उं टुं पट्टा रेखा</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>jēkar_tū [mai] [sujhāa] nahīmann ìā tām tūmpachtā ēgā</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&gt;{PRON,POS:null:mod}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
{N,ABS,^KAR,^TIME:+MODRES:null}

Action taken

It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘mod’ relation is resolved between two analysis windows as shown below.

mod(suggestion, I(icl>person))

Iteration:4

Node-net

[jēkar_tūṃ] [sujhāa] nahī ਮਨੀ ਤਾਮ ਤੂ ਪਛਤਾ ਏਗਾ

Rule fired

No rule fired.

Action taken

The analysis windows are shifted to right.

Iteration:5

Node-net

[jēkar_tūṃ] [sujhāa] nahī ਮਨੀ ਤਾਮ ਤੂ ਪਛਤਾ ਏਗਾ

Rule fired

No rule fired.

Action taken

The analysis windows are shifted to right.

Iteration:6

Node-net

[jēkar_tūṃ sujhāa [nahī] [mann] ਤਾਮ ਤੂ ਪਛਤਾ ਏਗਾ

Rule fired

>{NEG:null:null} {V:+.@not:null}

Action taken

It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes. Since, this attribute is preceded by ‘.@’ sign, it is concatenated to corresponding UW as UNL attributes. Here, the UW ‘agree(icl>do)’ is modified as ‘agree(icl>do).@not’.

Iteration:7

Node-net

[jēkar_tūṃ] [sujhāa] ਤਾਮ ਤੂ ਪਛਤਾ ਏਗਾ

Rule fired

No rule fired.
<table>
<thead>
<tr>
<th>Iteration</th>
<th>Node-net</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>ਜੇਕਰ_ਤੂ [ਸੁੱਖ] [ਤੂ] ਟੂ ਟੂ ਪਛਤਾ ਏਗਾ</td>
<td>The analysis windows are shifted to right.</td>
</tr>
<tr>
<td></td>
<td>jēkar_tū [sujhāa] [mann] Ṯ ਰ ਮਾ ਪਛਤਾ ਏਗਾ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rule fired</td>
<td>&gt; {N,INANI,MACH,WICH,PLC,RSN,SRCRESC,TOON,TAK,NOO,ben/pur/to,AOJ,plc/scn,ACT:NULL:obj} {V,WICH:OBJRES:NULL}</td>
</tr>
<tr>
<td></td>
<td>Action taken</td>
<td>It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘obj’ relation is resolved between two analysis windows as shown below. obj(agree(icl&gt;do),@not, suggestion)</td>
</tr>
<tr>
<td>9</td>
<td>ਜੇਕਰ_ਤੂ [ਸੁੱਖ] [ਤੂ] ਟੂ ਟੂ ਪਛਤਾ ਏਗਾ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>jēkar_tū [sujhāa] [mann] Ṯ ਰ ਮਾ ਪਛਤਾ ਏਗਾ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rule fired</td>
<td>&gt; {PRON,ANIMT,CCLB,CAG,NOO:NULL:agt} {V,VAUX,AGTRES,VOCURR,COO:+AGTRES,+CCLB,+@pred:NULL}</td>
</tr>
<tr>
<td></td>
<td>Action taken</td>
<td>It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows as shown below. agt(agree(icl&gt;do),@not,@pred, you(icl&gt;person))</td>
</tr>
<tr>
<td>10</td>
<td>ਜੇਕਰ</td>
<td>This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these</td>
</tr>
</tbody>
</table>
attributes are preceded by ‘@’ sign, they are concatenated to corresponding UW as UNL attributes. Here, the UW ‘agree(icl>do)’ is modified as ‘agree(icl>do).@not.@pred.@sg.@male’.

<table>
<thead>
<tr>
<th>Iteration: 11</th>
<th>Node-net</th>
<th>mann_īā [tā] tūṃ pachtā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>No rule fired.</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>The analysis windows are shifted to right.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 12</th>
<th>Node-net</th>
<th>mann_īā [tā] tūṃ pachtā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>-{CCL,^V,^N,^PRON,^ADJ,^ADV:null:null}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{PRON:+CCL:null}</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>This rule is preceded by ‘-’ sign, which indicates that it is a right composition rule. It results into concatenation of left node to the right node as a single composite node and the attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. This rule concatenates the adverb of condition delimiter, recognized by the presence of the attribute ‘CCL’ to the pronoun on the right. After the application of this rule, pronoun gets the attributes ‘CCL’ in addition of its existing attributes. When other relations are resolved with this pronoun, it will retain this attribute for further processing.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 13</th>
<th>Node-net</th>
<th>mann_īā [tā tūṃ] pachtā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>No rule fired.</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>The analysis windows are shifted to right.</td>
<td></td>
</tr>
<tr>
<td>Iteration:14</td>
<td>Node-net</td>
<td>Rule fired</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
</tr>
</tbody>
</table>
|             | मन्न_इंगा [उ_ं]   [पंड] देखा | >{PRON,ANIMT,CCL,^CAG,^NOO:null:agt}  
{V,^VAUX,^AGTRES,^VOCURR,^COO:+AGTRES,  
+CCL:null} | It is a right modification rule and results into deletion of left  
node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows.  
tagt(repent(icl>do), you(icl>person)) |
|             | मन्न_इंगा [उ_ं]   [पंड] देखा | >{V,CCLB:+.@entry:con}{V,CCL:null:null} | |
| Iteration:15| Node-net | Rule fired | Action taken |
|             | मन्न_इंगा [उ_ं]   [पंड] देखा | >{V,CCLB:+.@entry:con}{V,CCL:null:null} | This right modification rule deletes the left node from the node-list and resolves a relation between left analysis having a main verb with adverb clause beginner condition delimiter ‘CCLB’ attribute and right analysis window having sub-ordinate verb with adverb clause ending condition delimiter attribute ‘CCL’. Thus, a ‘con’ relation is resolved between main verb and sub-ordinate adverb clause of time. Due to presence of main clause and sub-ordinate clause in a UNL relation, a composite UW ‘:01’ is resolved by the system for UW ‘repent(icl>do)’ and it replaces UW ‘repent(icl>do)’ with ‘:01’ as shown below in resolved relation ‘con’. Here, the system also converts earlier resolved UNL relations involving ‘repent(icl>do)’ as composite relation as shown below.  
tagt:01(repent(icl>do), you(icl<person))  
con(agree(icl>do), @not.@pred.@sg.@male.@entry, :01) |
<table>
<thead>
<tr>
<th>Iteration:16</th>
<th>Node-net</th>
<th>[ਪ੍ਰਤਤਾ] [ਐਗਾ]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rule fired</td>
<td>+{V,m:sg:+.@future.@male:null}{{ਅਗਾ}:null:null}</td>
</tr>
<tr>
<td></td>
<td>Action taken</td>
<td>This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these attributes are preceded by ‘.@’ sign, they are concatenated to corresponding UW as UNL attributes. Here, the UW ‘repent(icl&gt;do)’ is modified as ‘repent(icl&gt;do).@future.@male’.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:17</th>
<th>Node-net</th>
<th>[ਪ੍ਰਤਤਾ_ਐਗਾ]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rule fired</td>
<td>No rule fired.</td>
</tr>
<tr>
<td></td>
<td>Action taken</td>
<td>Since, it is a last node so ‘.@entry’ is added to UW ‘repent(icl&gt;do).@future.@male’ and UNL generation process completes at this iteration.</td>
</tr>
</tbody>
</table>

After processing of input example sentence given in (4.60), the system generates the equivalent UNL expression as shown in (4.61).

{unl}
mod(suggestion, I(icl>person))
obj(agree(icl>do),@not.@pred.@sg.@male.@entry, suggestion)
agt(agree(icl>do),@not.@pred.@sg.@male.@entry, you(icl>person))
agt:01(repent(icl>do).@future.@sg.@male.@entry, you(icl<person))
con(agree(icl>do).@future.@sg.@male.@entry, :01)
{/unl}  
…(4.61)

Here, ‘:01’ indicates that it is a complex node. The UNL graph for UNL expression given in (4.61) is shown in Figure 4.7.
4.7.2.3 EnConversion process of sentences containing adverb clause for manner

The adverb clause for manner is resolved between the verb of main clause and the verb of sub-ordinate clause using the attribute ‘MCL’ (manner clause). The words ਜਦਾ ਅਮ jiddā Ṁ and ਦਾ ਅਮ ōdā Ṁ act as the clause delimiter for adverb clause for manner in Punjabi. The word ਜਦਾ jiddā Ṁ indicates the beginning of the manner condition and is represented with ‘MCLB’ attribute (manner clause beginning), while word ਦਾ ōdā Ṁ indicates the action as a result of that manner and is represented with ‘MCL’ attribute.

The handling of adverb clause of manner is illustrated with an example Punjabi sentence given in (4.62).

Example Punjabi sentence: ਜਦਾ ਤੂੰ ਕਹੇਂਗਾ ਦਾ ਮੈਂ ਦੌਂਗਾ।

Transliterated Punjabi sentence: jiddā Ṁ tū Ṁ kahēṅgā ōdā Ṁ mai Ṁ dauṅgā.

Equivalent English sentence: I will run as you will say.

The input sentence will be processed with parser phase, linked list creation phase, UW lookup phase, case marker lookup phase and unknown word handling phase of Punjabi EnConverter. After this, the node-net will be ready for UNL generation phase. The process of UNL generation phase for example sentence indicating node-net, rule fired and action taken by the fired rule at each iteration is depicted in Table 4.4. Here, left and right analysis windows are enclosed within ‘[’ and ‘]’.
Table 4.4: EnConversion process of adverb clause of manner for example sentence given in (4.62)

<table>
<thead>
<tr>
<th>Iteration: 1</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ਵਜੱਦਾਂ] [ਤੂੰ] ਕਵਹ ਏਗਾ ਓਦਾਂ ਮੈਂ ਦੌੜ ਏਗਾ</td>
<td>-{MCLB,^V:null:null}{PRON,^MCLB:+MCLB:null}</td>
<td>This rule is preceded by ‘-’ sign, which indicates that it is a right composition rule. It results into concatenation of left node to the right node as a single composition node and the attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. This rule concatenates the adverb of manner delimiter, recognized by the presence of the attribute ‘MCLB’ to the pronoun on the right. After the application of this rule, pronoun gets the attributes ‘MCLB’ in addition of its existing attributes. When other relations are resolved with this pronoun, it will retain this attribute for further processing.</td>
</tr>
<tr>
<td></td>
<td>[ਜੀਦਾ ਮੁਂ] [ਕਹੀ] ਏਗਾ ਓਦਾਂ ਮੈਂ ਦੌੜ ਏਗਾ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 2</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ਵਜੱਦਾਂ] [ਤੂੰ] ਕਵਹ ਏਗਾ ਓਦਾਂ ਮੈਂ ਦੌੜ ਏਗਾ</td>
<td>&gt;{PRON,ANIMT,MCLB,^CAG,^NOO,^PSP,^POS,^KI, ^TCLB,^TCL,^CCLB,^CCL:null:agt} {V,^AGTRES,^VOCURR,^COO:+AGTRES,+MCLB :null}</td>
<td>It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows as shown below and right analysis node, i.e., the main verb of the sentence gets ‘AGTRES’ and ‘MCLB’ attributes in addition of its existing attributes</td>
</tr>
<tr>
<td></td>
<td>[ਜੀਦਾ ਮੁਂ] [ਕਹੀ] ਏਗਾ ਓਦਾਂ ਮੈਂ ਦੌੜ ਏਗਾ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
for further processing.
agt(say(icl>do), you(icl>person))

<table>
<thead>
<tr>
<th>Iteration:3</th>
<th>Node-net</th>
<th>Rule fired</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ਕਵਹ] [ਏਗਾ] ਓਦਾਂ ਮੈਂ ਦੌੜ ਏਗਾ</td>
<td>+{V.m.sg:+.@future.@sg.@male:null}{[ਏਗਾ]:null:null}</td>
<td></td>
</tr>
<tr>
<td>[kahi] ēgā ōdā mai m dau ēgā</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these attributes are preceded by ‘@’ sign, they are concatenated to corresponding UW as UNL attributes. Here, the UW ‘say(icl&gt;do)’ is modified as ‘say(icl&gt;do).@future.@sg.@male’.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:4</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ਕਵਹ_ਐਗਾ] [ਓਦਾਂ] ਮੈਂ ਦੌੜ ਏਗਾ</td>
<td>No rule fired.</td>
<td>The analysis windows are shifted to right.</td>
<td></td>
</tr>
<tr>
<td>[kahi_ ēgā] [ōdā m] mai m dau ēgā</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This rule is preceded by ‘-’ sign, which indicates that it is a right composition rule. It results into concatenation of left node to the right node as a single composite node and the attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. This rule concatenates the adverb of manner delimiter, recognized by the presence of the</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
attribute ‘MCL’ to the pronoun on the right. After the application of this rule, pronoun gets the attributes ‘MCL’ in addition of its existing attributes. When other relations are resolved with this pronoun, it will retain this attribute for further processing.

| Iteration:6 | Node-net | कविंद्र एगाओ [हिंदी में] हैं [हिंदी में] हैं।
|             | Rule fired | No rule fired. |
|             | Action taken | The analysis windows are shifted to right. |

| Iteration:7 | Node-net | कविंद्र एगाओ [हिंदी में] हैं [हिंदी में] हैं।
|             | Rule fired | >[PRON,ANIMT,MCL,^CAG,^NOO,^PSP,^POS,^KI, ^TCLB,^TCL,^CCLB,^CCL:null:agt]
|             | Action taken | It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows as shown below and right analysis node, i.e., the verb of the sub-ordinate clause gets ‘AGTRES’ and ‘MCL’ attributes in addition of its existing attributes for further processing.
|             |             | agt(run(icl>do), l(icl>person)) |

| Iteration:8 | Node-net | कविंद्र एगाओ [हिंदी में] हैं [हिंदी में] हैं।
|             | Rule fired | >[V,MCLB:null:null]{V,MCL:+.@entry:man} |
|             | Action taken | This right modification rule deletes the left node from the node-list and resolves a ‘man’ relation between left analysis window having a main verb with adverb manner clause beginning delimiter ‘MCLB’ attribute and right
**Analysis Window**

Analysis window having sub-ordinate verb with adverb manner clause ending delimiter attribute ‘MCL’. Due to presence of main clause and sub-ordinate clause in a UNL relation, a composite UW ‘:01’ is resolved by the system for UW ‘run(icl>do)’ and it replaces UW ‘run(icl>do)’ with ‘:01’ as shown in resolved relation ‘man’. Here, the system also converts earlier resolved UNL relations involving ‘run(icl>do)’ as composite relation as shown below.

man(say(icl>do).@future.@sg.@male.@entry, :01) 
tag:01(run(icl>do).@future.@sg.@male.@entry, I(icl>person))

<table>
<thead>
<tr>
<th><strong>Iteration:</strong> 9</th>
<th><strong>Node-net</strong></th>
<th>[댔다] [화가]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[daut] [ēgā]</td>
<td></td>
</tr>
<tr>
<td><strong>Rule fired</strong></td>
<td>+{V,m,sg:+.@future.@sg.@male:null}{[ форум]:null:null}</td>
<td></td>
</tr>
</tbody>
</table>
| **Action taken** | This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these attributes are preceded by ‘.@’ sign, they are concatenated to corresponding UW as UNL attributes. Here, the UW ‘run(icl>do)’ is modified as ‘run(icl>do).@future.@sg.@male’.

<table>
<thead>
<tr>
<th><strong>Iteration:</strong> 10</th>
<th><strong>Node-net</strong></th>
<th>[EFAULT] [화가]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[daut_ ēgā]</td>
<td></td>
</tr>
<tr>
<td><strong>Rule fired</strong></td>
<td>No rule fired.</td>
<td></td>
</tr>
<tr>
<td><strong>Action taken</strong></td>
<td>Since, it is a last node so ‘.@entry’ is added to UW ‘run(icl&gt;do).@future.@sg.@male’ and UNL generation process completes at this iteration.</td>
<td></td>
</tr>
</tbody>
</table>
After processing of input example sentence given in (4.62), system generates equivalent UNL expression as given in (4.63).

\[
\begin{align*}
\text{agt}(\text{say}(\text{icl}>\text{do}).\text{future}.\text{sg}.\text{male}.\text{entry}, \text{you}(\text{icl}>\text{person})) \\
\text{agt:01}(\text{run}(\text{icl}>\text{do}).\text{future}.\text{sg}.\text{male}.\text{entry}, I(\text{icl}>\text{person})) \\
\text{man}(\text{say}(\text{icl}>\text{do}).\text{future}.\text{sg}.\text{male}.\text{entry}, :01)
\end{align*}
\]

UNL graph for this UNL expression is given in Figure 4.8.

4.7.2.4 EnConversion process of sentences containing adverb clause for place

The adverb clause for place is resolved between the verb of main clause and the verb of sub-ordinate clause using the attribute ‘PCL’ (place clause). The words ਜਿਥੇ jitthē and ਉੱਥੇ utthē act as the clause delimiters for adverb clause for place in Punjabi. The word ਜਿਥੇ jitthē indicates the beginning of place condition and is represented with ‘PCLB’ attribute (place clause beginning), while word ਉੱਥੇ utthē indicates the action at that place and is represented with ‘PCL’ attribute.

Handling of adverb clause for place is illustrated with an example Punjabi sentence given in (4.64).

Example Punjabi sentence: ਜਿਥੇ ਉਹ ਕਹੇਗਾ ਉੱਥੇ ਮੈਂ ਜਾਗਾ। …(4.64)

Transliterated Punjabi sentence: jitthē uh kahēgā utthē majāvārgā.

Equivalent English sentence: I will go wherever he will say.
The input sentence will be processed with parser phase, linked list creation phase, UW
lookup phase, case marker lookup phase and unknown word handling phase of Punjabi
EnConverter. After this, the node-net will be ready for UNL generation phase. The
process of UNL generation phase for example sentence indicating node-net, rule fired
and action taken by the fired rule at each iteration is depicted in Table 4.5. Here, left and
right analysis windows are enclosed within ‘[’ and ‘]’.

Table 4.5: EnConversion process of adverb clause for place for example sentence given
in (4.64)

<table>
<thead>
<tr>
<th>Iteration:1</th>
<th>Node-net</th>
<th>[jitthē] [uh] kahi ēgā utthē mai ṁ jā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>No rule fired.</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>The analysis windows are shifted to right.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:2</th>
<th>Node-net</th>
<th>[jitthē] [uh] [kahi] ēgā utthē mai ṁ jā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>&gt;{PRON,ANIMT,^CAG,^NOO,^PSP,^POS,^KI,^TCLB,^TCL,^CCLB,^CCL:null:agt} {V,^AGTRES,^VOCURR,^COO:+AGTRES:null}</td>
<td></td>
</tr>
</tbody>
</table>
| Action taken| It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows as shown below. 
agt(say(icl>do), he(icl>person)) |

<table>
<thead>
<tr>
<th>Iteration:3</th>
<th>Node-net</th>
<th>[jitthē] [kahi] ēgā utthē mai ṁ jā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>-{PCLB,^V:null:null} {V,^PCLB:+PCLB:null}</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>This rule is preceded by ‘-’ sign, which indicates that it is a right composition rule. It results into concatenation of left node to the right node as a single composite node and the attributes of right node are inherited for further processing.</td>
<td></td>
</tr>
</tbody>
</table>
This rule concatenates the adverb of place delimiter, recognized by the presence of the attribute ‘PCLB’ to the verb on the right. After the application of this rule, verb of the main sentence gets the attributes ‘PCLB’ in addition of its existing attributes. When other relations are resolved with this verb, it will retain this attribute for further processing.

<table>
<thead>
<tr>
<th>Iteration: 4</th>
<th>Node-net</th>
<th>वज़्हे कव़ह एगा उघे मैं जाएगा । जिठ्ठे_क्षी_एगा ut्थे mai mā jā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>+{V,m,sg:+.@future.@sg.@male:null}{[एगा]:null:null}</td>
<td></td>
</tr>
</tbody>
</table>
| Action taken | This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these attributes are preceded by ‘.@’ sign, they are concatenated to corresponding UW as UNL attributes. Here, the UW ‘say(icl>do)’ is modified as ‘say(icl>do).@future.@sg.@male’.

<table>
<thead>
<tr>
<th>Iteration: 5</th>
<th>Node-net</th>
<th>वज़्हे कव़ह एगा उघे मैं जाएगा । जिठ्ठे_क्षी_एगा ut्थे mai mā jā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>No rule fired.</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>The analysis windows are shifted to right.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 6</th>
<th>Node-net</th>
<th>वज़्हे कव़ह एगा उघे मैं जाएगा । जिठ्ठे_क्षी_एगा ut्थे mai mā jā ēgā</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule fired</td>
<td>-{PCL:null:null} {PERPRON,^PCL:+PCL:null}</td>
<td></td>
</tr>
<tr>
<td>Action taken</td>
<td>This rule is preceded by ‘-’ sign, which indicates that it is a right composition rule. It results into concatenation of left node to the right node as a single composite node and the</td>
<td></td>
</tr>
</tbody>
</table>
attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. This rule concatenates the adverb of manner delimiter, recognized by the presence of the attribute ‘PCL’ to the personal pronoun on the right. After the application of this rule, personal pronoun gets the attributes ‘PCL’ in addition of its existing attributes. When other relations are resolved with this personal pronoun, it will retain this attribute for further processing.

<table>
<thead>
<tr>
<th>Iteration: 7</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>jitthē_khi ēgā [utthē_mai] jā ēgā</td>
<td>No rule fired.</td>
<td>The analysis windows are shifted to right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 8</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>jitthē_khi ēgā [utthē_mai] [jā] ēgā</td>
<td>&gt;{PERPRON,ANIMT,PCL,CAG,NOO,PSP,POS,KI,TCLB,TCL,CCLB,CCL:null:agt}{{V,AGTRES,VOCURR,COO:AGTRES,PCL:null}}</td>
<td>It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows as shown below and right analysis node, i.e., the main verb of the sentence gets ‘AGTRES’ and ‘PCL’ attributes in addition of its existing attributes for further processing. agt(go(icl&gt;do), I(icl&gt;person))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration: 9</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>jitthē_khi ēgā [jā] ēgā</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

132
<table>
<thead>
<tr>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>{V,PCLB:+.@entry:null}{V,PCL:null:plc}</td>
<td>This right modification rule deletes the left node from the node-list and resolves a ‘plc’ relation between left analysis window having a main verb with adverb manner clause beginning delimiter ‘PCLB’ attribute and right analysis window having sub-ordinate verb with adverb manner clause ending delimiter attribute ‘PCL’. Due to presence of main clause and sub-ordinate clause in a UNL relation, a composite UW ‘:01’ is resolved by the system for UW ‘go(icl&gt;do)’ and it replaces UW ‘go(icl&gt;do)’ with ‘:01’ as shown in resolved relation ‘plc’. Here, the system also converts earlier resolved UNL relations involving ‘go(icl&gt;do)’ as composite relation as shown below. plc(say(icl&gt;do), :01) agt:01(go(icl&gt;do), I(icl&gt;person))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:10</th>
<th>Node-net</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ਜਾ] [ਏਗਾ]</td>
<td>[jā] [ēgā]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>{V.m.sg:+.@future.@sg.@male:null}{[ਏਗਾ]:null:null}</td>
<td>This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these attributes are preceded by ‘.@’ sign, they are concatenated to corresponding UW as UNL attributes. Here, the UW ‘go(icl&gt;do)’ is modified as ‘go(icl&gt;do).@future.@sg.@male’.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:11</th>
<th>Node-net</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ਜ_ਏਗਾ]</td>
<td>[jā_ēgā]</td>
</tr>
</tbody>
</table>

| Rule fired | No rule fired. |
Since, it is a last node so ‘@entry’ is added to UW ‘go(icl>do).@future.@sg.@male’ and UNL generation process completes at this iteration.

After processing of input example sentence given in (4.64), system generates equivalent UNL expression as given in (4.65).

\[
\{\text{agt}(\text{say}(\text{icl>do}).@\text{future.@sg.@male.@entry}, \text{he}(\text{icl>person})))
\]
\[
\text{agt:01(go}(\text{icl>do}).@\text{entry.@future.@sg.@male.@entry}, \text{I}(\text{icl>person}))
\]
\[
\text{plc(say}(\text{icl>do}).@\text{future.@sg.@male.@entry}, :01)
\]
\[
\{/\text{unl}\}
\]

The UNL graph for this UNL expression is given in Figure 4.9.

**Figure 4.9: UNL graph for UNL expression given in (4.65)**

4.7.3 EnConversion of sentences containing adjective clause

An adjective clause is described as a group of words that contains a subject and a predicate on its own and acts as the adjective for the subject in the main sentence (Dwivedi, 2002). Punjabi makes the use of form ਜੋ jo and ਉਹ uh structure in adjective clause sentence. It means that it has a form of ਜੋ jo followed by the subordinate clause. This subordinate clause is further followed by a corresponding form of ਉਹ uh in a sentence.

Handling of these type of sentences is illustrated with an example Punjabi sentence given in (4.66).
Example Punjabi sentence: ਵਜਸ ਬੱਚੇ ਨੂੰ ਰਾਮ ਨੇ ਬਚਾਇਆ। 

Transliterated Punjabi sentence: jis baccē nūṃ rām nē bacāiā uh sōhṇā sī.

Equivalent English sentence: The child that Ram saved was beautiful.

The input sentence will be processed with parser phase, linked list creation phase, UW lookup phase, case marker lookup phase and unknown word handling phase of Punjabi EnConverter. After this, the node-net will be ready for UNL generation phase. The process of UNL generation phase for example sentence indicating node-net, rule fired and action taken by the fired rule at each iteration is depicted in Table 4.6. Here, left and right analysis windows are enclosed within ‘[’ and ‘]’.

Table 4.6: EnConversion process of adjective clause example sentence given in (4.66)

<table>
<thead>
<tr>
<th>Iteration:1</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ਐ] [ਬੱਚਾ] ਨੂੰ ਰਾਮ ਨੇ ਬਚਾਇਆ। [ਐ] [baccā] nūṃ rām nē bacāiā uh sōhṇā sī</td>
<td>-{ADJCLB,^V: null: null}</td>
<td>This rule is preceded by ‘-’ sign, which indicates that it is a right composition rule. It results into concatenation of left node to the right node as a single composite node and the attributes of right node are inherited for further processing. The presence of ‘+’ sign in the action part of right analysis window results into the addition of attributes to the corresponding node. This rule adds the adjective clause beginner delimiter, recognized by the presence of the attribute ‘ADJCLB’ to the lexical semantic attribute list of right analysis window. After the application of this rule, noun gets the attributes ‘ADJCLB’ in addition of its existing attributes. When other relations are resolved with this noun, it will retain this attribute for further processing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration:2</td>
<td>Node-net</td>
<td>Rule fired</td>
<td>Action</td>
</tr>
<tr>
<td></td>
<td>[ਐ_ਬੱਚਾ] [ਐ] ਨੂੰ ਰਾਮ ਨੇ ਬਚਾਇਆ। [ਐ_baccā] nūṃ rām nē bacāiā uh sōhṇā sī</td>
<td>+{N: +NOO: null}</td>
<td>It is a left composition rule and results into concatenation</td>
</tr>
</tbody>
</table>
of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of ‘NOO’ attribute in lexical semantic attribute list of node at left analysis window.

<table>
<thead>
<tr>
<th>Iteration:3</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[जो बच्चा_नुं] [राम] बच्चा बिघा ठूंग मेहंडा मी</td>
<td>No rule fired.</td>
<td>The analysis windows are shifted to right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:4</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[जो बच्चा_नुं] [राम] बच्चा बिघा ठूंग मेहंडा मी</td>
<td>+{N,ANIMT,+CASE:null} {CASE:+null:null}</td>
<td>It is a left composition rule and results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of ‘CASE’ attribute in lexical semantic attribute list of node at left analysis window.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:5</th>
<th>Node-net</th>
<th>Rule fired</th>
<th>Action taken</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[जो बच्चा_नुं] [राम] बच्चा बिघा ठूंग मेहंडा मी</td>
<td>No rule fired.</td>
<td>The analysis windows are shifted to right.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration:6</th>
<th>Node-net</th>
<th>Rule fired</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[जो बच्चा_नुं] [राम] बच्चा बिघा ठूंग मेहंडा मी</td>
<td>&gt;{N,ANIMT,^CAG,^NOO,^PSP,^POS,^KI, ^TCLB,^TCL,^CCLB,^CCL:null:agt}</td>
</tr>
</tbody>
</table>
Action taken

It is a right modification rule and results into deletion of left node from the node-list, while the right node remains in the node-list. Here, ‘agt’ relation is resolved between two analysis windows as shown below.

\(\text{agt(save(icl>do), Ram(icl>person))}\)

Iteration: 7

Node-net

\([jō\_baccā\_nū]\ [ bacā]\ iā\ uh\ sōhnā\ sī\)

Rule fired

\(\{\text{N,NOO,ADJCLB,^TIME:+.@entry:obj}\}\)

\(\{\text{V:+OBJRES:null}\}\)

Action taken

In case of adjective clause sentence, object of the sub-ordinate clause acts as the entry node in a UNL expression. Thus, ‘+.@entry’ appears in the action part of left analysis window to add ‘@entry’ to the UW ‘child(icl>person)’. Here, ‘obj’ relation is resolved between two analysis windows as shown below.

\(\text{obj(save(icl>do), child(icl>person).@entry)}\)

Iteration: 8

Node-net

\([bacā]\ [ iā]\ uh\ sōhnā\ sī\)

Rule fired

\(\{\text{V,m,sg:+IYA,+.@sg.@male:null}\}\)
| Iteration:9 | Node-net | as ‘save(icl>do),@sg.@male’.

[bacā_īā] [uh] sōhṇā sī |

Rule fired: +{V,ADJCLB:+PRONADD:null}{PRON:null:null} |

| Action taken | This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of ‘PRONADD’ attribute in lexical semantic attribute list of node at left analysis window. |

| Iteration:10 | Node-net | |

[bacā_īā_ uh] [sōhṇā] sī |

Rule fired: >{V,ADJCLB,PRONADD:null:aoj}{ADJ:+.@entry:null} |

| Action taken | This right modification rule deletes left node from the node-list. Here, left analysis window has attributes ‘V’ (verb) and adjective clause beginning delimiter ‘ADJCLB’. Thus, it is a case of structure of जो jō followed by the subordinate clause. It has a right analysis window with ‘ADJ’ attribute, so a relation ‘aoj’ is resolved between two UWs. Due to presence of जो phrase with subordinate clause verb on left analysis window and ‘ADJ’ on right analysis window a composite UW ‘:01’ is resolved by the system for UW ‘save(icl>do)’ and UW ‘save(icl>do)’ is replaced with ‘:01’ as shown in resolved relation ‘aoj’. Here, the system also converts earlier resolved UNL relations involving ‘save(icl>do)’ as composite relation as shown below. |

| |

agt:01(save(icl>do),@sg.@male, Ram(icl>person)) |

obj:01(save(icl>do),@sg.@male, child(icl>person)) |
| Iteration:11 | Node-net | $[सोहना ] [सी]$
|             |          | $[sōhāṇā] [sī]$
| Rule fired  | +{ADJ:+.@past:null}{[[सी]:null:null]}
| Action taken| This left composition rule results into concatenation of right node to the left node as a single composite node and the attributes of left node are inherited for further processing. The presence of ‘+’ sign in the action part of left analysis window results into the addition of attributes. Since, these attributes are preceded by ‘.@’ sign, they are concatenated to corresponding UW as UNL attributes. Here, the UW ‘beautiful(icl>state)’ is modified as ‘beautiful(icl>state).@past’.
| Iteration:12 | Node-net | $[सोहना ] [सी]$
|             |          | $[sōhāṇā ] [sī]$  
| Rule fired  | No rule fired
| Action taken| Since it is a last node so ‘.@entry’ is added to UW ‘beautiful(icl>state).@past’ and UNL generation process completes at this iteration.

After processing of input example sentence given in (4.66), system generates its equivalent UNL expression as shown in (4.67).

{unl}
agt:01(save(icl>do).@sg.@male.@past, Ram(icl>person))
obj:01(save(icl>do).@sg.@male.@past, child(icl>person).@entry)
aoj(beautiful(icl>state).@past.@entry, :01)
{/unl} 

The UNL graph for this UNL expression is given in Figure 4.10. Here, scope node ‘:01’ has an ‘aoj’ relation with the entry node. Here, the entry node of scope node’s UNL graph is ‘child(icl>person)’ acts as the child node of relation ‘obj’ for parent node
‘save(icl>do)’, indicating that it is an adjective clause sentence and asserts that central focus is on the ‘child(icl>person)’ and not on the action ‘save(icl>do)’.

Figure 4.10: UNL graph for UNL expression given in (4.67)
The above mentioned strategies empower Punjabi EnConverter to successfully convert simple as well as complex Punjabi sentences into their equivalent UNL expressions.
Punjabi EnConverter converts an input Punjabi sentence into its equivalent UNL expression. It processes a given input sentence from left to right using analysis window and condition window. It also uses EnConversion analysis rules including left composition rules (+), right composition rules (-), left modification rules (<), right modification rules (>) and attribute changing rules (:) during the conversion of input Punjabi sentence to its equivalent UNL expression. Punjabi EnConverter uses Punjabi shallow parser for processing the input Punjabi sentence. This parser performs the tasks of tokenizer, morph analyzer, part-of-speech tagger and chunker for the processing of input sentence. Architecture of Punjabi EnConverter has seven phases, namely, parser phase (to parse the input sentence with Punjabi shallow parser), linked list creation phase, universal word lookup phase, case marker lookup phase, unknown word handling phase, user interaction phase (this phase is optional) and UNL generation phase. Punjabi EnConverter can convert simple and complex Punjabi sentences to equivalent UNL expressions.

There are three basic types of subordinate clauses in a complex sentence that have been considered here. These are: noun clause, adjective clause and adverb clause. The noun clause is resolved between the verb of main clause and the verb of subordinate clause. Punjabi EnConverter has used an attribute ‘NCL’ (noun clause) for identifying noun clause delimiter. Adverb clause is a sub-ordinate clause that acts as adverb in a sentence. In order to resolve UNL relations for adverb clauses, these are classified as adverb clauses for time, for condition, for manner and for place. The adverb clause for time uses ‘TCLB’ and ‘TCL’ attributes as clause delimiters, adverb clause for condition uses ‘CCLB’ and ‘CCL’ attributes as clause delimiters, adverb clause for manner uses ‘MCLB’ and ‘MCL’ attributes as clause delimiter while adverb clause for place uses ‘PCLB’ and ‘PCL’ attributes as clause delimiters. An adjective clause is described as a group of words that contains a subject and a predicate on its own and acts as the adjective for the subject in the main sentence. In this chapter, the process of EnConversion of Punjabi sentences including these clauses has been explained, in detail. These details are supported by illustrations of process with the help of example Punjabi sentences.