UNL Based Machine Translation System for Punjabi Language

A Thesis

Submitted in the fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

Submitted by

Parteek Kumar

(Registration No. 90603501)

Under the supervision of

Dr. R. K. Sharma

Professor

School of Mathematics and Computer Applications,

Thapar University, Patiala

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

THAPAR UNIVERSITY, PATIALA-147004 (Punjab), INDIA

February 2012

Certificate

I hereby certify that the work which is being submitted in this thesis entitled "UNL Based Machine Translation System for Punjabi Language", in fulfillment of the requirements for the award of the degree of DOCTOR OF PHILOSOPHY submitted in Department of Computer Science and Engineering, Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. R.K. Sharma and refers work of other researchers which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other University.

Parteek Kumar

(Parteek Kumar)
Regd. No. 90603501

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge and belief.

R.K. Sharma

(R.K. Sharma)
Professor, School of Mathematics and Computer Applications,
Thapar University, Patiala-147 004 (INDIA)
Supervisor

Acknowledgements

With the grace of God, it is very special moment to acknowledge the contributions of those who supported me during the journey to achieve this task. This research work is the outcome of very valuable guidance from my supervisor Dr. R.K. Sharma. He has always been there to guide me towards the right direction whenever I got stuck with the ideas. I want to thank him for his valuable evenings and week-ends that he spent with me in finalizing the writings of this thesis work. I also want to thank Mrs. Sharma to provide a conducive environment in her home during discussions on this proposed work.

The motivation of this research work is provided by Dr. Pushpak Bhattacharyya, Prof. Department of Computer Science and Engineering, IIT Bombay. He introduced me to the concept of UNL. He is very kind to provide me the required NLP resources and guidance. I will always remain indebted to him for showing me the path of development of Punjabi EnConverter and DeConverter.

I express my gratitude to the Doctoral Committee for monitoring the progress and providing valuable suggestions for improvement of my research work. I feel very proud and thankful to the Department of Computer Science and Engineering, Thapar University, Patiala for providing all the resources for a good research work. I am very thankful to the administration of Thapar University for granting me a teaching off for one semester. This period helped me immensely to compile my research work. I am also very thankful to Dr. Seema Bawa, Dr. Maninder Singh and all my colleagues for their constant support and cooperation during my research work.

I take this opportunity to say thanks to my family members, who are always there to support me morally and emotionally. With the grace of God, this work provides me the opportunity to make my parents; Mr. Ved Kumar, Mrs. Jagdish Bhatia, S. Dalip Singh and Mrs. Joginder Kaur feel proud. I want to thank my wife Mrs. Sanmeet Kaur, kids Rahat and Rishan for providing a very loving, cool and calm environment in our home.

Parteek Kumar

(Parteek Kumar)

Abstract

Machine Translation (MT) has been an area of immense interest among researchers during last couple of decades. This area has witnessed a few lows and highs during its life span and has also witnessed integration of research works from different fields including linguistics, computer science, artificial intelligence, statistics, mathematics, philosophy and others. Researchers have proposed different paradigms for machine translation across natural languages with reasonable success. Universal Networking Language (UNL) based MT is also an effort in this direction. The UNL programme was launched in 1996 in Institute of Advanced Studies (IAS) of United Nations University (UNU), Tokyo, Japan. The approach in UNL revolves around the development of an EnConverter and a DeConverter for a natural language. The EnConverter is used to convert a given sentence in natural language to an equivalent UNL expression; and the DeConverter is used to convert a given UNL expression to an equivalent natural language sentence. In the work carried out in this PhD project, these two components, namely, EnConverter and DeConverter have been developed for Punjabi language.

The PhD thesis on the work carried out in this project is divided into seven chapters. These chapters are: Introduction, Review of Literature; UNL Framework and Creation of Punjabi-Universal Word Lexicon; Punjabi-UNL EnConverter; UNL-Punjabi DeConverter; Results and Discussions; Conclusion and Future Scope of the work.

First chapter contains introduction to Machine Translation and its need in the age of Information Technology. In this chapter, the challenges of MT, approaches of MT, objectives of this research, methodology adopted to achieve these objectives and the features of Punjabi language have been presented. Machine Translation approaches that can be classified into four categories, namely, direct MT, rule-based MT, corpus-based MT and knowledge-based MT, have also been discussed in this chapter. The main objective of this study is to design and develop a multi-lingual machine translation system for Punjabi language. A major outcome of this research work is the development of Punjabi EnConverter, Punjabi DeConverter and a web interface for online EnConversion and DeConversion task.

Second chapter of the thesis provides the details of the findings of research in the area of MT. The review of literature on MT has been performed by tracking the historic developments in this field. The literature reported in this work has been so organised as to include the state of MT before the invention of computers, beginning of automated MT (1946-1954), decade of high expectations and disillusion (1955-1966), post ALPAC decade (1967-1976), the revival of MT research (1977 to 1989), decade of 1990 to 2000, research since 2000 and MT for Indian languages and research activities in UNL.

Computerized translation was first performed by Georgetown Automatic Translation (GAT) system at Georgetown University, USA. During the period of 1955-1966, RAND Corporation's statistical analysis of a large corpus of Russian Physics texts, *'CETA'* a hybrid system, Mechanical Translation and Analysis of Languages (METAL) with the use of Chomsky's transformational paradigm, and *'Logos'* machine translation system had been some of the important developments in this area. During the post ALPAC period (1967-1976), *'Ariane'* system was developed and *'TAUM'* project was undertaken by using syntactic transfer for English-French translation. The revival of research on MT took place during 1977-89 with *'Ariane'* system that was developed on linguistics based transfer approach. Other important MT projects such as *'ATLAS'* system, *'UNITRAN'* system, IBM's *'Candide'* system, Universal Networking Language (UNL) based MT system, *'KANTOO'* system, *'ALEPH'* a pure example-based machine translation system, *'SisHiTra'* a hybrid Machine Translation system, Google Translation and *'OpenLogos'* system have also been discussed in this chapter.

The review of literature on MT for Indian languages has also been presented in this chapter. The important machine translation systems for Indian languages like *'AnglaBharti'*, *'AnuBharati'*, *'Anusaaraka'*, *'MANTRA'* (MAchiNe assisted TRAnslation tool) system, English-Bangla-ANUBAD system, a translation system for bi-lingual Hindi-English (Hinglish) text, *'Shakti'* system, *'MaTra'* system, a Punjabi to Hindi MT system, English to Urdu translation system, and *'Sampark'* a hybrid system for translation among Indian languages have been discussed in this chapter.

The research activities in UNL have been presented in three distinct sections in this chapter. These sections are: development of EnConversion and DeConversion modules; applications of UNL in other contexts; and use of external lexical and ontological

resources to enhance some of the processes of UNL. The work on conversion of Brazilian Portuguese into UNL and vice-versa, EnConversion and DeConversion tools for Tamil language, *'HERMETO'* system, French EnConverter and French DeConverter, UNL DeConverter for Chinese language, *'Manati'* DeConversion model, UNL-Nepali DeConverter, UNL-Hindi DeConverter, Arabic MT system based on UNL and a Bangla EnConversion system have also been reviewed in this chapter.

Third chapter discusses UNL format for information representation that includes the details on UWs and their four types (Basic UWs, Restricted UWs, Extra UWs and Temporary UWs), UNL relations, UNL attributes and formats to write UNL sentence. Compound UWs in UNL are used to denote compound concepts that are to be interpreted as a whole so that one can use their parts at the same time. This chapter also provides the details on UNL system that consists of EnConverter, DeConverter, Dictionary Builder, Grammar Rules, UNL Key Concept in Context (KCIC), UW Gate Universal Parser, UNL Verifier, Language Server and Word Dictionary (Language-UW Dictionary). UNL system makes use of word dictionaries in the form of Language-UW lexicon of respective languages for its processing. An entry of the word dictionary contains three parts, namely, a headword, a UW and a set of morphological, syntactic and semantic attributes. This chapter discusses the grammatical attributes of Punjabi-UW lexicon, important issues in creation of Language-UW dictionary and creation of Punjabi-UW dictionary. A Punjabi-UW dictionary having 1,15,000 entries has been developed in this work by taking Hindi-UW dictionary as a reference.

Fourth chapter of this thesis provides the details of EnConversion system for conversion of input Punjabi sentences into UNL. This chapter also discusses the framework for designing the EnConverter for Punjabi language with a special focus on generation of UNL attributes and relations from Punjabi source text. The architecture of Punjabi EnConverter has been divided into seven phases. These phases are: (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) and (vii) UNL generation phase. These phases have been implemented using Java for the development of proposed system. All these phases are illustrated by giving example sentences.

Fifth chapter discusses UNL to Punjabi DeConverter that generates natural language Punjabi sentence from a given input UNL expression. The architecture of Punjabi DeConverter has been divided into five phases, namely, (i) UNL parser phase, (ii) Lexeme selection phase, (iii) Morphology generation phase, (iv) Function word insertion phase and (v) Syntax planning phase. The first stage of a DeConverter is UNL parser which parses the input UNL expression to build a node-net from the input UNL expression. During lexeme selection phase, Punjabi root words and their dictionary attributes are selected for the given UWs in the input UNL expression from the Punjabi-UW dictionary. After that, the nodes are ready for generation of morphology according to the target language in the morphology phase. The proposed system makes use of morphology rule base for Punjabi language to handle attribute label resolution morphology; relation label resolution morphology; and noun, adjective, pronoun and verb morphology. In function word insertion phase, the function words are inserted to the morphed words. These function words are inserted in the generated sentence based on nine column rule base. Finally, the syntax planning phase is used to define the word order in the generated sentence so that output matches with a natural language sentence.

The pseudocodes and algorithms for building data structures in UNL parser; processing of noun morphology and adjective morphology rule base; processing of function word insertion rule base; controlling the syntax planning of nodes of simple UNL graph; syntax planning of UNL graph with a scope node; handling of untraversed parent node and nodes with multiple parents nodes during syntax planning, handling of some special cases of syntax planning and syntax planning of noun, adjective and adverb clause sentences have been presented in this chapter. All these pseudocodes and algorithms have been implemented in Java to develop the proposed UNL-Punjabi DeConverter.

Sixth chapter of this thesis contains the results and discussion on the work done in this project. The evaluation of proposed system, consisting of Punjabi EnConverter and DeConverter, has been performed with the help of one thousand Punjabi sentences. These sentences have been selected in such a way that generation of all possible UNL relations and attributes can be tested. For testing purpose, Spanish UNL Language Server and agricultural domain threads developed by IIT Bombay, India are considered as gold-standards.

Spanish Language Server contains English sentences with their corresponding UNL expressions generated by the system (Spanish Language Center, 2004) while agricultural domain threads developed by IIT Bombay have Hindi language sentences with their equivalent UNL expressions. These sentences were translated manually into equivalent Punjabi sentences and then inputted to the proposed Punjabi EnConverter system for their EnConversion to UNL. The UNL expression generated by proposed system is compared with the UNL expression given by the gold-standard EnConverter. These two UNL expressions match with each other if the UNL relations, including associated UWs and UNL attributes present in the expressions are same. It has been seen that proposed system handles the resolution of UNL relations and generation of attributes for these sentences with a very reasonable accuracy. Proposed Punjabi DeConverter is also evaluated by inputting UNL expressions generated by Punjabi EnConverter to it and the output of Punjabi DeConverter is compared with input Punjabi sentence given to Punjabi EnConverter. The two Punjabi sentences (original sentence inputted to Punjabi EnConverter and the sentence generated by DeConverter) are compared and the system is evaluated based on adequacy test and BLEU score.

Subjective tests like adequacy and fluency tests have been performed on the proposed system. BLEU score has been calculated to evaluate the quality of output. The quantitative test, namely, error analysis has also been performed by calculating Sentence Error Rate (SER) and Word Error Rate (WER). From this analysis, it has been concluded that proposed system generates 89.0% intelligible sentences and generates 92.0% sentences that are faithful to the original sentences. The system could achieve a fluency score of 3.61 (on a 4-point scale) and adequacy score of 3.70 (on a 4-point scale). The proposed system is able to achieve a BLEU score of 0.72. The proposed system has a word error rate of 5.43% and sentence error rate of 20.8%. These scores of the proposed system can be improved further by improving the rule base and lexicon.

Chapter seven presents the conclusion, limitations and future scope of work in order to refine the proposed development of EnConverter and DeConverter for Punjabi language.

List of Figures

List of Tables

List of Algorithms

List of PseudoCodes

Contents

Chapter 1
Introduction

Homo sapiens have a great tendency of getting their work done by machines and as a result of that they have invented a number of machines. The invention of computers in previous century has resulted into opening of a number of research areas. Machine Translation (MT), also known as automatic translation or mechanical translation, is an area of research that has attracted many researchers during last couple of decades. MT includes the computerized methods that automate all or part of the translation process from one natural language to another. This area has witnessed a few lows and highs during its life span and has also witnessed integration of research works from different fields including linguistics, computer science, artificial intelligence, statistics, mathematics, philosophy and others. Researchers have proposed different paradigms, like direct MT, rule-based MT (transfer-based and interlingua-based), corpus-based MT and knowledge-based MT.

Universal Networking Language (UNL) based MT (developed on interlingua-based approach) is also an effort in this direction. The UNL programme was launched in 1996 in Institute of Advanced Studies (IAS) of United Nations University (UNU), Tokyo, Japan and it is currently supported by Universal Networking Digital Language (UNDL) foundation, an autonomous organization. The approach in UNL revolves around the development of an EnConverter and a DeConverter for a natural language. The EnConverter is used to convert a given sentence in natural language to an equivalent UNL expression; and the DeConverter is used to convert a given UNL expression to an equivalent natural language sentence. UNL system has the potential to bridge the language barriers across the world with the developments of $2n$ components, while traditional approaches requires the $n*(n-1)$ components, where n is the number of languages. UNL represents the information at sentence level in the form of Universal Words (UWs), UNL relations and UNL attributes. The concepts are represented by UWs and UNL relations are used to specify the role of each word in a sentence. The subjective

meaning of the sentence is expressed through UNL attributes. UNL system makes use of word dictionary that stores that information in the form of root words of the language and the corresponding UWs. In the work carried out in this PhD thesis, two components, namely, EnConverter and DeConverter have been developed for Punjabi language.

Punjabi language can be written using two scripts, namely, *Gurmukhi* and *Shahmukhi* script. *Gurmukhi* script is used in eastern Punjab (India), and *Shahmukhi* script in western Punjab (Pakistan). The proposed work handles the Punjabi sentences written in *Gurmukhi* script. The examples given in this thesis work are in *Gurmukhi* script along with their transliteration and gloss in *Roman* script. For inline examples, transliteration is provided in italics and gloss is provided in italics with in single quotes, *e.g.*, ਮਾਤਾ *mātā 'mother'*. The transliteration provided in this thesis is based on *Gurmukhi* to *Roman* transliteration software *'GTrans'*, developed by Punjabi University, Patiala, India. The UNL graphs presented in this thesis are developed by using UNL graphs web verifier and visualizer tool developed by Spanish Language Center (2004).

In this chapter, the need and challenges of machine translation, approaches of machine translation, objectives of this research, methodology adopted to achieve these objectives and the features of Punjabi language have been presented.

1.1 Need of Machine Translation

MT has a great social and commercial importance. The following points highlight the need of MT in today's world.

1.1.1 Socio-Political importance of MT

There are communities in the world where more than one language is generally spoken. The dominance of one language may prove disadvantageous to the speakers of other language and may even cause the disappearance of a language. The loss of a language often results into the disappearance of a distinctive culture, and a way of thinking. Thus, MT becomes a social and political necessity for modern societies, who do not wish to impose a common language on their members. It is also a necessity of organizations like the United Nations and the European Community, for whom multilingualism is both a basic principle and a fact of everyday life (Arnold *et al.*, 1994).

1.1.2 Commercial importance of MT

MT has become a necessity in the scenario of global economy. Translation of user guides and user interfaces in multiple languages is the key of success for multi-national companies. Manual translation is slow and proves to be costly owing to the requirement of highly skilled manpower (Arnold *et al.*, 1994).

1.1.3 Coverage of literary works by MT

MT can break the language barriers by making the availability of rich sources of literature to people across the world in their native language. It is really amazing to think of an MT system that can translate literary works from any language into our native language (Siddiqui and Tiwary, 2008).

1.1.4 MT to bridge the digital divide

MT can help to overcome technological barriers. Internet has a great influence on the working of human beings these days. One can find vast amount of information on Internet. The usage of this information is beyond the reach of a significant portion of society as most of this information is in English. MT can help in bridging the digital divide, by translating web pages and electronic mail messages into the native language of a user (Siddiqui and Tiwary, 2008).

1.1.5 MT to assist human translator

Online versions of electronic dictionaries and translation systems can assist human translators in the process of translation. These systems combine multilingual word processing, OCR facilities, terminology management software and translation memories to facilitate human translators. The translation memories enable translators to store original texts and their translated versions side by side. The translator can search here for phrases or even full sentences in one language and can get corresponding phrases in the other language (Hutchins, 2003).

1.1.6 MT for cross language information retrieval

In multilingual environment, there is a need of information retrieval systems capable of searching text in many languages. Cross Language Information Retrieval (CLIR) deals with retrieving information written in a language that is different from the language of the user's query. For example, a user may pose his query in Punjabi and can retrieve relevant documents written in English. CLIR makes use of MT in two ways: it uses MT system to

translate foreign language documents into the language of the user's query and it translates the user's query into target language. The target language query is then used to retrieve target language documents using classical information retrieval techniques (Siddiqui and Tiwary, 2008).

Thus, MT has a potential to overcome language barriers and to make communication between users of different languages much easier.

1.2 Challenges of Machine Translation

There are number of challenging issues in MT that make it a difficult problem to solve. Some of the important issues are structural differences between languages, ambiguity, multiword units *etc.* In this section, some of the issues are presented in brief.

1.2.1 Word order

The arrangement of words in a sentence varies across languages. For example, English language is Subject-Verb-Object (SVO) type language whereas most Indian languages, including Punjabi, are Subject-Object-Verb (SOV) type language. This makes the approach of word by word translation impractical.

1.2.2 Lexical differences

Sometimes, a word used in one language has no single-word equivalent in another language which results into lexical differences between languages. The example sentence given in (1.1) illustrates lexical differences between English and Punjabi.

English sentence: Ram hugged Rahat. …(1.1)

Equivalent Punjabi sentence: ਰਾਮ ਨੇ ਰਾਹਤ ਨੂੰ ਗਲ਼ੇ ਲਗਾਇਆ । …(1.2)

Transliterated Punjabi sentence*: rām nē rāhat nūṃ gḷē lagāiā.*

Here, *'hug'* does not have a single-word equivalent in Punjabi; it has 'ਗਲ਼ੇ ਲਾਉਣਾ' '*gḷē lāuṇā'* as multiple-words equivalent in Punjabi.

1.2.3 Lexical ambiguity

If a word has more than one meaning, then it is classified as lexically ambiguous. The example Punjabi sentences given in (1.3), (1.4) and (1.5) illustrates concept of lexical ambiguity.

Punjabi sentence: ਕੱਪੜੇ ਵਿਚ ਵੱਟ ਹਨ । …(1.3)

Transliterated sentence: *kapp̣ē vic va̱ṯ han.*

Equivalent English sentence: There are wrinkles in the cloth.

Punjabi sentence: ਇਹ ਵੱਟ ਸਾਡੇ ਖੇਤ ਵਿਚ ਹੈ । …(1.4)

Transliterated sentence: *ih vaṭṭ sāḍē khēt vic hai.*

Equivalent English sentence: This mud path is in our field.

Punjabi sentence: ਮੈਂ ਰੱਸੀ ਵੱਟ ਰਹੀ ਹਾਂ । …(1.5)

Transliterated sentence: *maiṃ rassī vaṭṭ rahī hāṃ.*

Equivalent English sentence: I am winding the rope.

Here, ਵੱਟ *vaṭṭ* can be replaced by *'wrinkle', 'mud path'*, or *'wind'* depending on the sense implied in the structure. The correct sense must first be identified for each of the words before selecting the appropriate replacement (Bharati *et al.*, 1994).

1.2.4 Structural ambiguity

A phrase or sentence can be interpreted in more than one way. This kind of ambiguity is known as structural ambiguity. This ambiguity is especially challenging because it requires a deep understanding of the speaker's intention, and we can often not be certain of what exactly the speaker meant. The sentence structure must be interpreted correctly for translation. The example English sentence given in (1.6) illustrates this concept.

I saw Ram on the hill with the telescope. …(1.6)

Here, the *'telescope'* could have been the instrument of seeing, or *'Ram'* could have been carrying the *'telescope'*, or it is the *'hill'* with *'telescope'*. Hence, it would be important to identify the relationship of the telescope correctly (Bharati *et al.*, 1994).

1.2.5 Pronoun resolution

The unresolved references of pronoun may lead to incorrect translation. The example English sentence given in (1.7) illustrates this concept.

A dog saw a cow on the road. It started barking on seeing it. …(1.7)

The first *'it'* in the example sentence given above can refer to a *'dog'*, *'cow'* or *'road'*. For its translation into Punjabi, the gender information about the referent will be important because the gender of the verb depends on it. If *'it'* was referring to the *'dog'* then the gender will be masculine but for the *'road'* and *'cow'* it will be feminine (Bharati *et al.*, 1994). So, it is very important to resolve *'dog'* or *'cow'* for *'it'* in the

example sentence given in (1.7).

1.2.6 Idioms and phrases

It is difficult to translate a sentence with idiomatic expressions, because idioms are composed of words that do not directly contribute to their meaning (Siddiqui and Tiwary, 2008). The example English sentence given in (1.8) illustrates this concept.

The old man finally kicked the bucket. …(1.8)

If the system does not recognize the idiom *'kicked the bucket'* which means *'to die'* then, its translation in Punjabi will end up as given in (1.9), which is a non-sense translation.

ਬੁੱਢੇ ਆਦਮੀ ਨੇ ਆਖਰ ਬਾਲ਼ਟੀ ਨੂੰ ਲੱਤ ਮਾਰੀ । …(1.9)

buḍḍhē ādmī nē ākẖar bāḷṭī nūṃ latt mārī.

These are some of the important challenges that one faces in the development and implementation of a machine translation system.

1.3 Translation Architecture

The various approaches of machine translation can be described with the help of Vauquois triangle (Jurafsky and Martin, 2000) as shown in Figure 1.1. In this triangle, vertical direction (height of the triangle) indicates the increasing depth of analysis and the horizontal direction (width of the triangle) indicates the amount of effort required for transfer. It is evident from Figure 1.1 that the base of the triangle needs the most transfer and the least analysis and generation, while the top of the triangle needs least transfer and most analysis and generation.

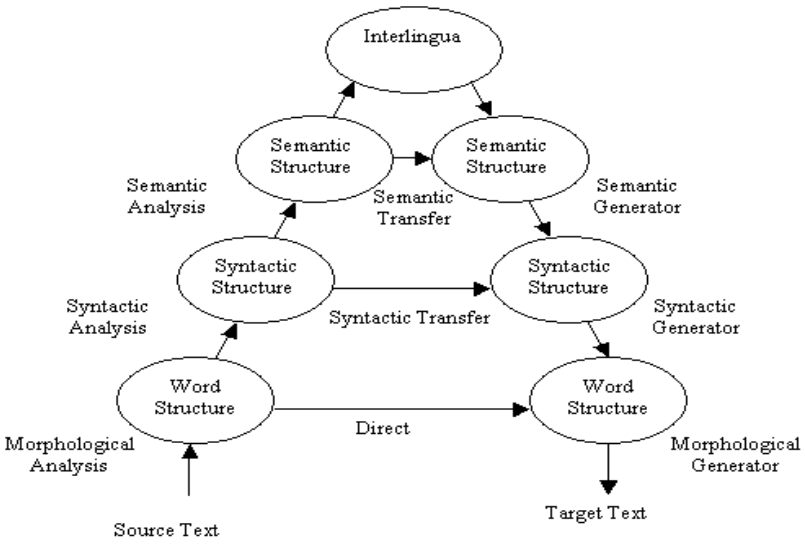

Figure 1.1: Vauquois triangle

In the next section, a brief detail about different machine translation approaches has been provided.

1.4 Machine Translation approaches

Machine Translation approaches can be classified into four categories, namely, direct MT, rule-based MT, corpus-based MT and knowledge-based MT, as depicted in Figure 1.2. The rule-based approach can further be classified into transfer-based approach and interlingua approach. The corpus-based translation approach can also further be classified as statistical machine translation and example-based machine translation. In this section, these approaches are briefly described.

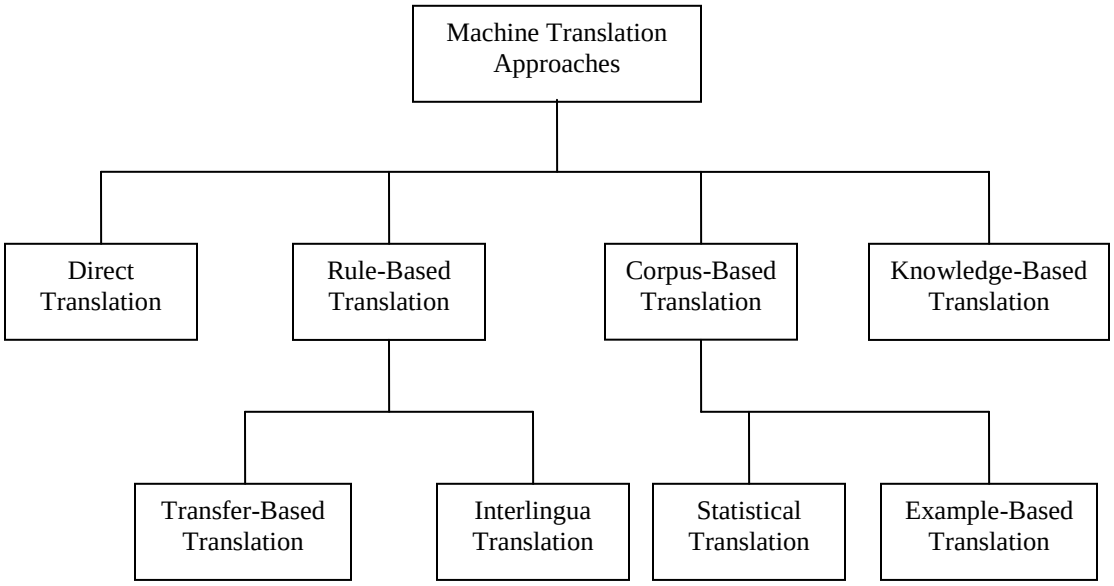

Figure 1.2: Machine Translation approaches

1.4.1 Direct MT

Direct MT technique was developed during 1950s to make use of newly invented computers for MT. It is based on a straightforward and easily implementable technique, keeping in view less processing power of computers available at that time. This method of translation is depicted in Figure 1.3.

A direct translation system carries out word-by-word translation with the help of bilingual dictionary. As such, it is also known as dictionary driven machine translation approach. It involves a parser, which performs preliminary analysis of the source language sentence to produce its parts of speech information. This information is processed by a rule base to transform the source language sentence into a target language

sentence. These rules include bilingual dictionary rules and rules to re-order the words. The direct machine translation system with parser and rule-base is also known as Transformer. The transformer architecture is shown in Figure 1.4 (Arnold *et al.*, 1994).

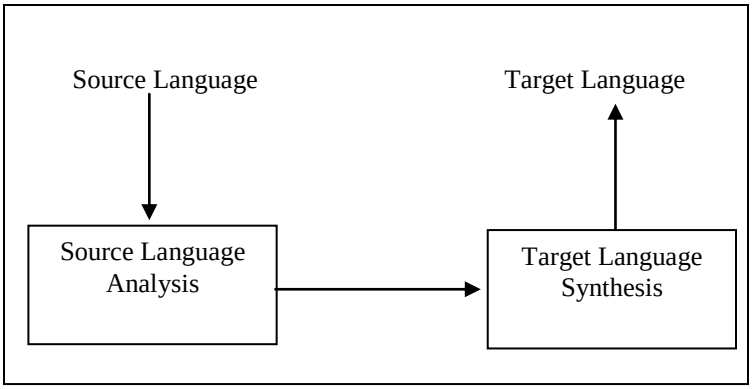

Figure 1.3: Direct translation method

The example English sentence given in (1.10) illustrates the concept of direct machine translation approach.

Amar slept in the garden. …(1.10)

To translate this sentence into Punjabi, a direct translation system will first look up a dictionary to get target language words for each word appearing in the source language sentence. Then, the words are re-ordered to match the default sentence structure of the target language, *i.e.*, SOV in case of Punjabi. The output of these steps is given in (1.11) and (1.12).

Word-by-word translation:

ਅਮਰ ਸੁੱਤਾ ਵਿਚ ਬਾਗ਼ …(1.11)

amar suttā vic bāġ

Syntactic rearrangement:

ਅਮਰ ਬਾਗ਼ ਵਿਚ ਸੁੱਤਾ …(1.12)

amar bāġ vic suttā

Translation among Indian languages is not very complex owing to the fact that most Indian languages have similar sentence format. Hence, a simple word-by-word

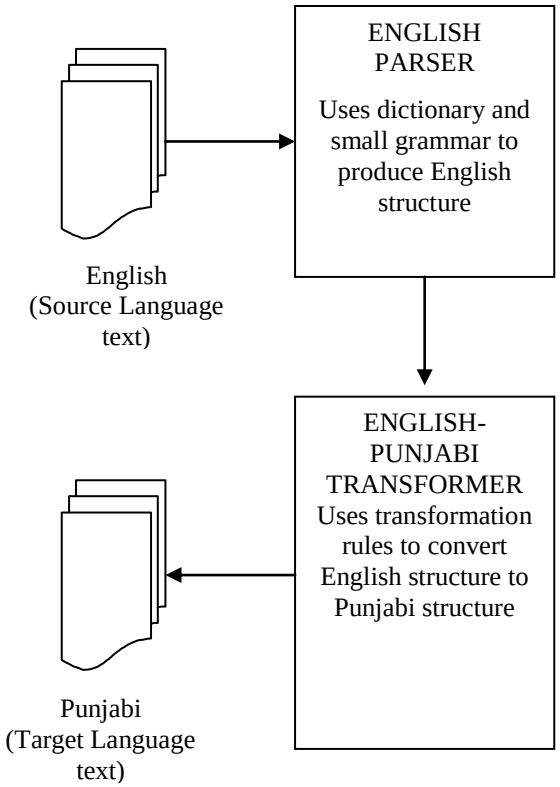

Figure 1.4: Transformer architecture

translation, with small set of rules may lead to an acceptable translation in many cases as shown in the example of Hindi-Punjabi translation given in (1.13) and (1.14).

Example Hindi source sentence:

राम ने राहत को प्यार से गले लगाया । …(1.13)

rām nē rāhat kō payār sē galē lagāyā.

Equivalent English sentence:

Ram hugged Rahat with love.

Equivalent Punjabi translated sentence:

ਰਾਮ ਨੇ ਰਾਹਤ ਨੂੰ ਪਿਆਰ ਨਾਲ ਗਲ਼ੇ ਲਗਾਇਆ । …(1.14)

rām nē rāhat nūm̐ piār nāl gl̤ē lagāiā.

The unidirectional nature of direct machine translation system is its major disadvantage. In a multilingual scenario, this may be quite expensive. For example, in order to provide translation capability for *n* number of languages, we need to develop *n**(*n*-1) MT systems (Siddiqui and Tiwary, 2008). Direct machine translation is not suitable for translation of complex sentences, because it requires complex grammatical analysis and word ordering rules. This approach is also not suitable for ambiguous sentences (Seasly, 2003).

1.4.2 Rule-Based MT system

The rule-based MT is used to remove major shortcomings of direct machine translation system. It parses the source text and produces an intermediate representation, which may be a parse tree or some abstract representation. The target language text is generated from the intermediate representation. These systems rely on specification of rules for morphology, syntax, lexical selection, semantic analysis, transfer and generation process. Due to the extensive use of rule-base, these systems are known as Rule-based systems. Depending on the intermediate representation used, these systems are further categorized as transfer-based machine translation and interlingua machine translation.

1.4.2.1 Transfer-Based MT

Transfer approach occupies the level above direct translation in the Vauquois triangle as shown in Figure 1.1. It is also known as Indirect or Linguistic Knowledge (LK) translation. It is used to capture the meaning of the original sentence to produce an intermediate representation. It has three intermediate stages in translation process that

include an analysis stage: to analyze the source text to produce source structure, a transfer stage: to transfer source structure to target structure and synthesis stage: to generate target language text from target structure as shown in Figure 1.5 (Arnold *et al.*, 1994).

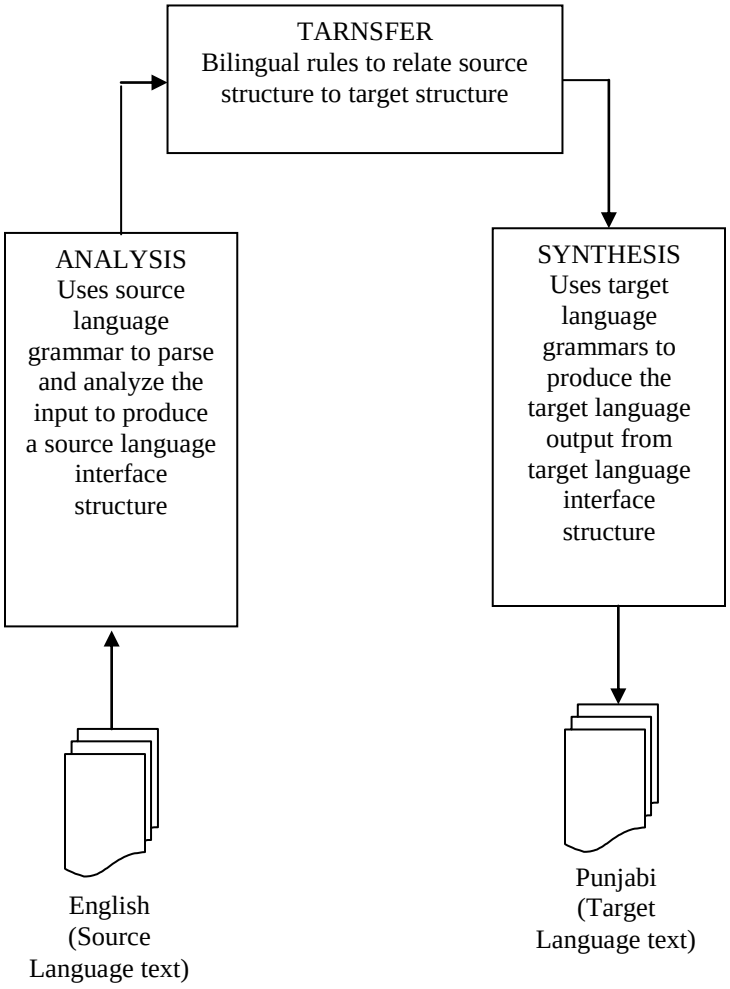

Figure 1.5: Transfer-based architecture

An advantage of this approach is its modular structure. Analysis of source language text is independent of language generator. All language-pair specific differences are captured in the transfer stage. To provide translation capability among multiple languages, we need an analyzer and a generator component for each language and a transfer component for each pair of such languages. For example, to provide translation capability for five languages, we need five analyzers, five generators, and twenty transfer components, while in direct translation approach 20 complete translation systems are required (Siddiqui and Tiwary, 2008).

Transfer approach requires a large number of rules for its implementation in multilingual machine translation systems. This is a bottleneck of such a system (Arnold *et al.*, 1994).

1.4.2.2 Interlingua-Based MT

The interlingua approach appears at the apex of the Vauquois triangle as shown in Figure 1.1. It is inspired by Chomsky's findings that regardless of varying surface syntactic structures, languages share a common deep structure. In interlingua-based MT approach, the source language text is converted into a language independent meaning representation

called Interlingua. As stated by Jurafsky and Martin (2000) ‘An interlingua represents all sentences that mean the same thing in the same way regardless of the source language they happen to be in’.

Interlingua-based MT system, involves two stages in the translation process, including analysis stage: to deeply analyze the source sentence for producing a language independent representation (interlingua); and synthesis stage: the target language is generated from the interlingua. This concept is shown in Figure 1.6. The system involves extensive use of source and target grammar rules during this process.

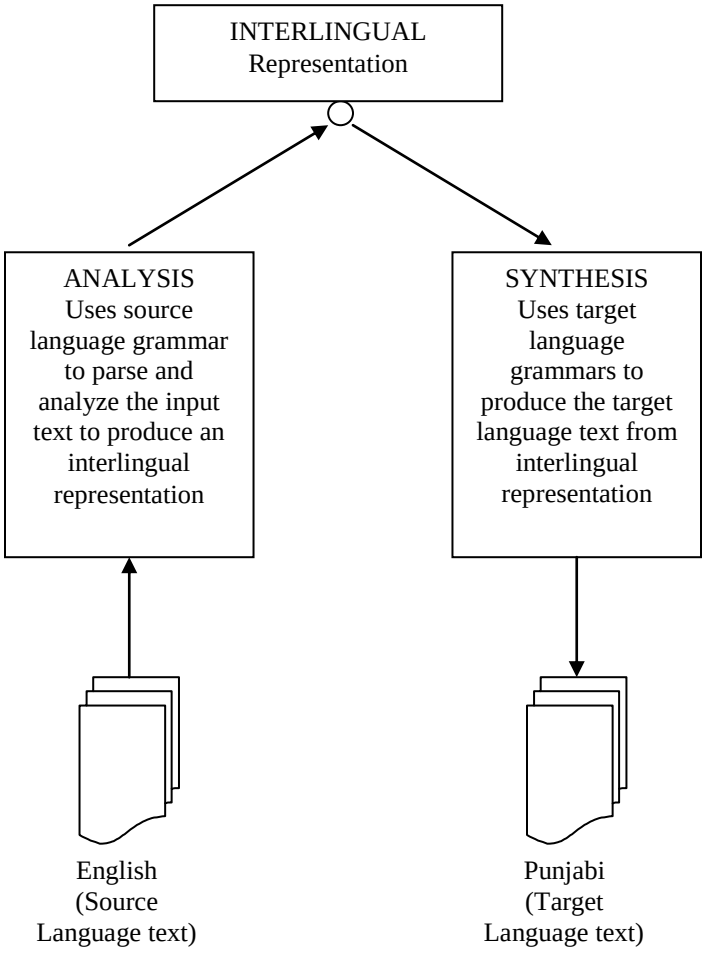

Figure 1.6: Architecture of interlingua system

Interlingua approach is best suited for multilingual machine translation, as it requires only two components for each language: one for conversion from source language to interlingua and other for interlingua to target language. As such, in this approach we require 2*n* components as shown in Figure 1.7 (Subramanian and Narayanan, 1990). The meaning based representation known as interlingua makes this approach suitable for information retrieval applications as well.

There are few problems with interlingua approach. The analysis of source text requires a deep semantic analysis, which sometime results into the loss of information. Another problem is the creation of an adequate interlingua, because it should be both abstract and independent of the source and target languages, which make this task difficult in case of

large multilingual machine translation systems involving the languages with wider differences.

1.4.3 Corpus-Based MT

Corpus-based MT systems have become popular in recent years. These are fully automatic systems that require significantly less human labor than traditional rule-based approaches. However, they require sentence aligned parallel text for each language pair and cannot be used for language pairs for which such corpora do not exist. The corpus-based approach is further classified into statistical and example-based machine translation approaches (Siddiqui and Tiwary, 2008).

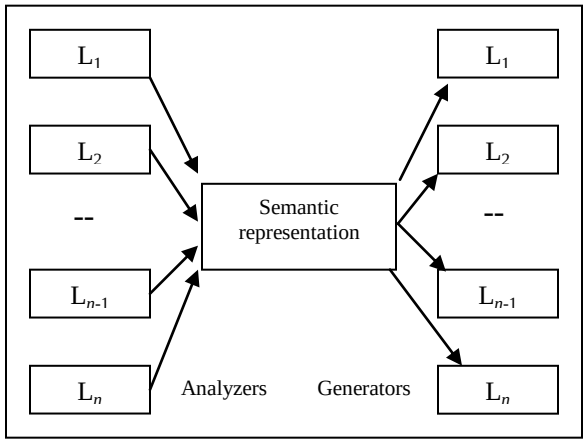

Figure 1.7: Interlingua-based translation among *n* languages

1.4.3.1 Statistical MT

Statistical Machine Translation (SMT) uses statistical models for translation whose parameters are derived from the analysis of bilingual text corpora. It does not make use of linguistic rules. SMT was introduced by Warren Weaver in 1949. SMT was re-introduced in 1991 by researchers at IBM. The essence of this method is first to align phrases, word groups and individual words of the parallel texts, and then calculate the probabilities that any one word in a sentence of one language corresponds to a word or words in the translated sentence with which it is aligned in other language. SMT has given more acceptable results by picking the word(s) that has the highest probability of occupying its current position, given the surrounding words (Seasly, 2003).

SMT considers the MT as a noisy channel metaphor process. It means that, if a user wants to translate a given sentence *'f'* in the source language *'F'* to a sentence *'e'* in the target language *'E'*, the noisy channel model handles this situation in the following way:

Suppose that the sentence *'f'* to be translated was initially conceived in language *'E'* as some sentence *'e'*. During communication, *'e'* was corrupted by the channel to *'f'*. Now, a user will assume that each sentence in *'E'* is a translation of *'f'* with some probability,

and the sentence that a user chooses as the translation (*'ê'*) is the one that has the highest probability as shown in mathematical expression given in (1.15) (Brown *et al.*, 1990).

$$\hat{e} = \arg_e \max Q(e|f) \quad \ldots(1.15)$$

Here, $Q(e|f)$ depends on two factors, (i) the kind of sentences that are likely in the language *'E'*. This is known as the Language Model (LM) and is represented mathematically as $Q(e)$, (ii) the way sentences in *'E'* get converted to sentences in *'F'*. This is called the Translation Model (TM) and represented mathematically as $Q(f|e)$. Thus, by applying Bayes' theorem in the expression given in (1.15), will result the equivalent mathematical expression as given in (1.16).

$$\hat{e} = \arg_e \max \frac{Q(e)Q(f|e)}{Q(f)} \quad \ldots(1.16)$$

Since, *'f'* is fixed, $Q(f)$ can be omitted from the expression given in (1.16) to obtain its equivalent mathematical expression given in (1.17) (Ramanathan, 2008).

$$\hat{e} = \arg_e \max Q(e)Q(f|e) \quad \ldots(1.17)$$

The concept of SMT is illustrated with an example English sentence given in (1.18).

He is walking. …(1.18)

The possible Punjabi translations of this example sentence are given in (1.19), (1.20) and (1.21).

ਉਹ ਚੱਲ ਰਿਹਾ ਹੈ । …(1.19)

uh call rihā hai.

ਚੱਲ ਰਿਹਾ ਹੈ ਉਹ । …(1.20)

call rihā hai uh.

ਉਹ ਤੁਰ ਰਿਹਾ ਹੈ । …(1.21)

uh tur rihā hai.

The expression given in (1.17) helps to select the appropriate translation of the source sentence. Because, though $Q(f|e)$ would be the same for the three sentences, the language model would rule out the last two sentences given in (1.20) and (1.21), *i.e.*, the first translation given in (1.19) would receive a much higher value of $Q(e)$ than the other two sentences. This leads to another perspective on the statistical MT model, *i.e.*, the best translation is the sentence that is both faithful to the original sentence and fluent in the

target language (Jurafsky and Martin, 2000). Thus, in the expression given in (1.17), $Q\ e'$ represents fluency and $Q\ f|e'$ represents faithfulness.

The open source tools like CMU-Statistical language modeling toolkit and *'SRILM'* are used for the creation of LM; *'GIZA++'* and *'MGIZA'* are used for the creation of TM; *'Moses'* and *'ISI ReWrite'* are used for creation of decoder in the process of SMT.

1.4.3.2 Example-Based MT

The Example-Based Machine Translation (EBMT) approach was suggested by Makoto Nagao in 1984. The EBMT approach requires a bilingual corpus with parallel texts. This approach works on the principle of translation by analogy. This principle is encoded in EBMT through example translations. This concept has been illustrated with an example of sample bilingual corpus of English-Punjabi as shown in Table 1.1.

Table 1.1: Example bilingual English-Punjabi corpus

English corpus	Punjabi corpus
For how much is that red umbrella?	ਲਾਲ ਛਤਰੀ ਕਿੰਨੇ ਦੀ ਹੈ? *lāl chtarī kinnē dī hai*?
For how much is that book?	ਕਿਤਾਬ ਕਿੰਨੇ ਦੀ ਹੈ? *kitāb kinnē dī hai*?

The sample corpus given in Table 1.1 makes it simpler to learn translations of sub-sentential units. After getting trained on sample corpus, the system learns the correspondences as given in Table 1.2.

Table 1.2: Correspondences learned by system

English sub-sentential units	Correspondence to Punjabi
For how much is that 'X'?	X ਕਿੰਨੇ ਦੀ ਹੈ? *X kinnē dī hai*?
Red umbrella	ਲਾਲ ਛਤਰੀ *lāl chtarī*
Book	ਕਿਤਾਬ *kitāb*

By composing these units, the system can produce novel translations in the future. The architecture of EBMT system is shown in Figure 1.8. An EBMT system has two main

modules, namely, retrieval and adaptation. The retrieval module is used to retrieve translation examples from example-base or Translation Memory (TMEM) for a given input and adaptation is used to carry out necessary modifications in the retrieved example pair to generate translation of target language sentence. The modification may involve addition, deletion, and replacement of morphological words.

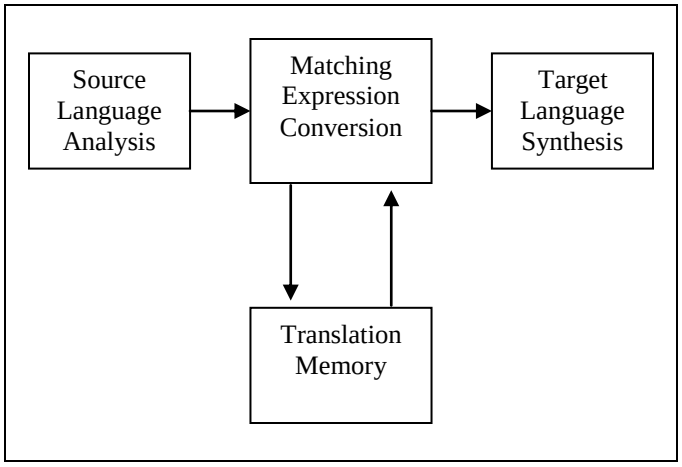

Figure 1.8: Architecture of EBMT

EBMT system has certain advantages over rule-based and SMT systems. In the rule-based system, knowledge about the syntax and semantics of source and target, need to be represented as rules, which are difficult to write. The problem with SMT systems is that they need a huge, aligned, parallel corpus. The availability of huge corpora is scarce. Therefore, SMT approach cannot be widely used. EBMT systems require neither a large set of rules, nor a huge parallel corpus. It only requires an example-base, which can be easily created from resources like multi-lingual official notices and reports (Siddiqui and Tiwary, 2008).

1.4.4 Knowledge-Based MT

The important process in knowledge-based translation is to capture as much linguistic knowledge as possible from the source language sentences and store this into the translation system's knowledge base. For this, the system makes the use of source and target language dictionaries; source and target language structures and rules; word meanings in different contexts and language constructs; domain specific terminology; previously translated words, phrases, sentences, paragraphs; ontological and lexical knowledge; language style and cultural differences *etc.* By capturing all these knowledge sources, the system produces a high quality output. It is implemented on the interlingua architecture, but differs from interlingual technique by the depth with which it analyzes the source language and its reliance on explicit knowledge of the world. It makes use of augmenter as shown in architecture of KBMT given in Figure 1.9. The only problem of

KBMT is that it is quite expensive to produce such a system because it requires a large amount of knowledge (Seasly, 2003).

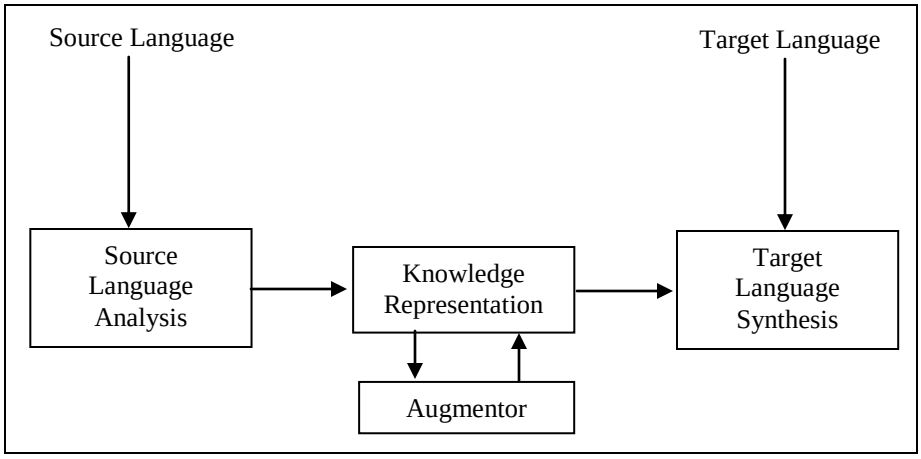

Figure 1.9: Architecture of KBMT

1.4.5 Hybrid approaches

The experiences of researchers in last couple of decades have shown that it is not possible to produce highly accurate system by relying on a single machine translation approach. Thus, nowadays, researchers are using hybrid approaches to improve the quality of existing systems. Most commonly, it involves the use of a linguistic method to parse the source text, and a non-linguistic method, such as statistical-based or example-based, to assist with finding the proper interpretation (Siddiqui and Tiwary, 2008).

1.5 Need of UNL based machine translation system for Punjabi language

There is an immense need to develop a machine translation system for Punjabi language. Some machine translation systems for Punjabi Language are at developing stages but they are limited only to a pair of languages, *i.e.*, Hindi-Punjabi. In multi-lingual environment, it is very important to have machine translation systems for large number of languages. In order to have a multi-lingual machine translation system for Punjabi language, it would require a separate system for every language, *i.e.*, $n*(n-1)$ language systems will be required for n number of languages. As such, there is a need to develop an internationally standardized system in which fewer components are needed for a multi-lingual machine translation. An interligua approach based machine translation system is one solution for this problem. UNL is an internationally standardized interligua that has the potential to bridge the language barriers across all the languages of the world.

Such a system for Punjabi language will certainly be very helpful for more than 91 million Punjabi language users (Lewis and Paul, 2009).

1.6 Objectives of this research

The main objective of this study is to design and develop a multi-lingual machine translation system for Punjabi language. In order to perform this task, following objectives were proposed to be carried out.

i) An electronic Punjabi-UW dictionary of about 40,000 root words was to be created in the proposed work.

ii) A DeConverter for Punjabi language was to be developed which can convert a UNL document to corresponding Punjabi language text.

iii) An EnConverter for Punjabi language was to be developed which can convert input Punjabi text to corresponding UNL output.

iv) A web interface was to be designed with the provisions for input of Punjabi sentences for corresponding UNL output and for input of a UNL document for corresponding Punjabi text output.

v) The scope of work was proposed to be limited to simple sentences with an approximate accuracy of 70%.

1.7 Major contributions and achievements

A major outcome of this research work is the development of Punjabi EnConverter and Punjabi DeConverter. A Punjabi EnConverter has been developed that converts input Punjabi sentence to corresponding UNL expression. A web interface has been designed for online EnConversion of input Punjabi sentence to UNL expression. It enables the conversion of input Punjabi sentence to other languages having their own UNL DeConverter. A Punjabi DeConverter has also been developed that converts a UNL expression to corresponding Punjabi language text. A web interface has also been designed for online DeConversion of UNL expression to corresponding Punjabi sentence. It enables the Punjabi readers to read the sentences in their local language that are originally written in different languages having their equivalent UNL expression present on the web.

This system will also provide an opportunity to research scholars working on MT to explore UNL as an interlingua.

1.8 Features of Punjabi language

Punjabi language is an Indo-Aryan language and is one of the constitutionally recognized languages of India. Indo-Aryan languages form a subgroup of the Indo-Iranian group of languages, which in turn belong to Indo-European family of languages. Punjabi is widely spoken in north-west India, Pakistan, United States, Australia, United Kingdom and Canada. There are more than 91 million native speakers of Punjabi language, which makes it approximately the 12th most widely spoken language in the world (Lewis and Paul, 2009). Gill (2008) has explained the features of Punjabi language. The following paragraphs give a comprehensive review of these features.

Punjabi has word classes in the form of noun, pronoun, adjective, cardinal, ordinal, main verb, auxiliary verb, adverb, postposition, conjunction, interjection and particle. Punjabi nouns change forms for number (singular or plural) and case in sentences. Punjabi nouns have assigned gender (masculine or feminine). For example, ਕੰਧ *kandh 'fence'*, ਕੁਰਸੀ *kurasī 'chair'*, ਸੜਕ *saṛak 'road' etc.* are used in feminine gender, and ਮੇਜ਼ *mēz 'table'*, ਟਰੱਕ *ṭarakk 'truck'*, ਦਿਨ *din 'day' etc.* are used in masculine gender.

Punjabi has six types of pronouns. These are: personal pronouns, *e.g.*, ਮੈਂ maiṃ *'i'*, ਤੂੰ tūṃ *'you'*; reflexive pronouns, *e.g.*, ਆਪ *āp* (some what equivalent to honorific form of English second person *'you'*); demonstrative pronouns, *e.g.*, ਉਹ *uh 'that'* and ਇਹ *ih 'this'*; indefinite pronouns, *e.g.*, ਕੋਈ *kōī*, ਕੁਝ *kujh*, ਸਾਰੇ *sārē etc.*; relative pronouns (to join two clauses in a complex sentence), *e.g.*, ਜੋ *jō* and ਜਿਹੜਾ *jihṛā* and interrogative pronouns, *e.g.*, ਕੌਣ *kauṇ 'who'*, ਕੀ *kī 'what'etc.*

In Punjabi language, adjectives usually precede the nouns but follow the pronouns. For example, ਸੋਹਣਾ *sōhṇā 'handsome'*, ਕਾਲਾ *kālā 'black'* are functioning as adjectives and they precede nouns in 'ਸੋਹਣਾ ਮੁੰਡਾ' *'sōhṇā muṇḍā' 'handsome boy'*, 'ਕਾਲਾ ਘੋੜਾ' *'kālā ghōṛā' 'black horse'*, respectively. The examples of adjectives following pronouns are, 'ਮੈਂ ਸੋਹਣਾ ਹਾਂ' *'maiṃ sōhṇā hāṃ' 'I am handsome'*, 'ਤੂੰ ਸੋਹਣਾ ਹੈਂ' *'tūṃ sōhṇā haiṃ' 'you are handsome'*. Punjabi adjectives can be classified into two categories, inflected and uninflected. For example, ਸੋਹਣਾ *sōhṇā 'handsome'* inflections are ਸੋਹਣਾ *sōhṇā 'handsome'* (direct singular), ਸੋਹਣੇ *sōhṇē 'handsome'* (oblique singular and direct plural),

and ਸੋਹਣਿਆਂ *sōhṇiāṃ ‘handsome’* (oblique plural). The uninflected adjectives are ਲਾਲ *lāl ‘red’*, ਮਿਹਨਤੀ *mihnatī ‘hardworking’*, ਮਸ਼ਹੂਰ *mashhūr ‘famous’ etc.*

Cardinals and Ordinals can be used for both the genders and change forms for case (direct and oblique). Except, ਇੱਕ *ikk ‘one’*, which is in singular number, all the remaining cardinals (ਦੋ *dō ‘two’*, ਪੰਜ *pañj ‘five’*, ਅਠਾਰਾਂ *aṭhārāṃ ‘eighteen’ etc.*) are in plural number. Generally, all the ordinals are used in singular number. For example, ਪੰਜਵਾਂ *pañjvāṃ ‘fifth’*, ਛੇਵਾਂ *chēvāṃ ‘sixth’ etc.* are all ordinals.

In a Punjabi sentence, verbs must agree with the subject or object of the sentence in terms of gender, number, and person. Punjabi verbs change forms for gender, number, person, and tense. The verbs have assigned transitivity and causality. In Punjabi, there are two auxiliary verbs – ਹੈ *hai* for present tense (*e.g.,* ਰਾਮ ਅੰਬ ਖਾਂਦਾ ਹੈ । *rām amb khāndā hai. ‘Ram eats mango’*) and ਸੀ *sī* for past tense (*e.g.,* ਰਾਮ ਨੇ ਅੰਬ ਖਾਧਾ ਸੀ । *rām nē amb khādhā sī. ‘Ram had eaten mango’*). All the forms of these two auxiliary verbs can equally be used for both the genders. For future tense in sentences, *‘EGA’* form of main verb is used and in those sentences auxiliary verb is thus not used (*e.g.,* ਰਾਮ ਅੰਬ ਖਾਵੇਗਾ । *rām amb khāvēgā. ‘Ram will eat mango’*).

Adverbs can indicate manner, time, place, condition *etc.* For example, ਉੱਪਰ *uppar ‘upon’*, ਉੱਤੇ *uttē ‘over’*, ਹੇਠਾਂ *hēṭhāṃ ‘below’ etc.* are some Punjabi adverbs. Postpositions are similar to prepositions in English. These link noun, pronoun, and phrases to other parts of the sentence. Some Punjabi postpositions are ਨੇ *nē*, ਨੂੰ *nūṃ*, ਉੱਤੇ *uttē ‘over’*, ਦਾ *dā ‘of’ etc.* In Punjabi, postpositions follow the noun or pronoun unlike English, where these precede the noun or pronoun, and thus termed prepositions. In Punjabi, the postpositions can be classified into two types, namely, inflected postpositions and uninflected postposition. For example, ਦਾ *dā ‘of’* (marker of possessive case) postposition is an inflected postposition because it changes forms for gender, number and case. There are a large number of postpositions, which do not change forms at all, *e.g.,* ਨੇ *nē* (instrumental or agentive case marker), ਨੂੰ *nūṃ* (generally used with objects), and ਤੋਂ *tōṃ ‘from’* are known as uninflected postpositions.

Conjunctions are used to join words, phrases, or independent clauses in compound sentences. For example, ਅਤੇ/ਤੇ *atē/tē 'and'*, ਜਾਂ *jāṃ 'or'* are acting as co-ordinate conjunctions. Sub-ordinate conjunctions are typically used to introduce dependent clauses in complex sentences. For example, ਜੇ *jē*, ਉਂਞ *uññ*, ਤਾਂ *tāṃ etc*.

Interjection is used to express emotions in sentences. For example, ਹਾਏ *hāē*, ਆਹਾ *āhā*, ਸਦਕੇ *sadkē*, 'ਜੀ ਆਇਆਂ ਨੂੰ' '*jī āiāṃ nūṃ' 'welcome' a*ll are interjections. There are various particles that are used in the Punjabi sentences for emphasis, negation *etc.* These particles do not change forms for any of the grammatical categories. These include, emphatic particles that are used to emphasize or put stress on some part of the sentences, *e.g.*, ਈ *ī*, ਹੀ *hī*, ਤਾਂ *tāṃ*, ਤੇ *tē*, ਖਾਂ *khāṃ*, ਵੀ *vī*, ਭੀ *bhī etc.*; negative particles that are normally used for negation effect in sentences, *e.g.*, ਨਹੀਂ *nahīṃ*, ਨਾ *nā*, ਨਾਹ *nāh*; honorific particles that are used for giving respect, *e.g.*, ਜੀ *jī*, ਜੀਓ *jīō*, ਸਾਹਿਬ *sāhib etc.* and vocative particles that are commonly used in vocative case, *i.e.*, to call someone, *e.g.*, ਅੜਿਆ *aṛiā* (masculine singular), ਅੜਿਓ *aṛiō* (masculine plural), ਅੜੀਏ *aṛīē* (feminine singular) *etc.*

Punjabi phrases can be broadly classified into two types, namely, nominal phrases (built using the words of various word classes like noun, pronoun, adjective *etc.*) and verb phrases (built using primarily the words of main verb and auxiliary verb word classes) (Gill, 2008).

1.9 Organization of thesis

This PhD thesis is divided into seven chapters. The details about the literature review carried out for this work is presented in Chapter 2. Introduction to UNL framework and creation of Punjabi-Universal Word lexicon has been discussed in Chapter 3. The details on the design and development of Punjabi-UNL EnConverter have been given in Chapter 4. Chapter 5 covers the design and development of UNL-Punjabi DeConverter. The results and discussions on the implementation of proposed work have been provided in Chapter 6. Chapter 7 of thesis discusses the conclusion and future scope of the work.

Chapter Summary

In this chapter, the need and challenges of machine translation, approaches of machine translation, objectives of this research, methodology adopted to achieve these objectives and the features of Punjabi language have been presented. The machine translation has socio-political and commercial importance in the modern world. It helps to provide wide readership to literary works and used to bridge the digital divide. It is also used for cross language information retrieval. Automated Machine Translation is a complex problem involving the challenges of word order, lexical differences, lexical ambiguity and structural ambiguity among the languages. Machine Translation approaches can be classified into four categories, namely, direct MT, rule-based MT, corpus-based MT and knowledge-based MT. A direct translation system carries out word-by-word translation with the help of bilingual dictionary. Direct machine translation is not suitable for translation of complex sentences and for multilingual translations. The rule-based MT is further categorized as transfer-based machine translation and interlingua machine translation. A transfer-based machine translation system has three intermediate stages in translation process that include an analysis stage**,** a transfer stage and synthesis stage. Interlingua-based MT system, involves two stages in the translation process, including analysis stage and synthesis stage**.** The corpus-based approach is classified into statistical and example-based machine translation. Statistical machine translation requires three key components, namely, Language Model (LM), Translation Model (TM) and decoder. Example-Based Machine Translation (EBMT) approach requires a bilingual corpus with parallel texts. An EBMT system has two main modules, namely, retrieval and adaptation. The main process in knowledge-based translation is to capture as much linguistic knowledge as possible from the source language sentences and store this into the translation system's knowledge-base. Nowadays, researchers are using hybrid approaches to improve the quality of existing systems.

The main objective of this study is to design and develop a multi-lingual machine translation system for Punjabi language. A major outcome of this research work is the development of Punjabi EnConverter, Punjabi DeConverter and a web interface for online EnConversion and DeConversion task.

Chapter 2
Review of Literature

2.1 Introduction

Development of translating systems using machines has been a long cherished dream of human beings. People had been working for the realization of this dream even before the invention of computers. The works of Georges Artsrouni and Petr Troyanskii are noteworthy in this direction. Researchers have proposed different approaches for developing a translation system. These approaches include, direct or dictionary-based approach, rule-based approaches (transfer and interlingua), knowledge-based approach and corpus-based approaches (example-based and statistical-based). Hutchins (1995) has mentioned that the differences between direct and indirect, transfer and interlingua, knowledge-based and corpus-based approaches are becoming less useful for the categorization of the translation systems. He has argued that transfer systems incorporate interlingual features (for certain areas of vocabulary and syntax); interlingua systems include transfer components; rule-based systems make increasing use of probabilistic data and stochastic methods; statistics and example-based systems include traditional rule-based grammatical categories and features. Following this argument, we have reviewed the literature of machine translation based in this chapter on the historic developments in the field. This chapter also contains the review of literature on the machine translation amongst Indian languages and efforts carried out in the area of Universal Networking Language (UNL).

2.2 State of Machine Translation before the invention of computers

The earlier works in the field of mechanized translation are by Georges Artsrouni and Petr Troyanskii. Both of them patented their works in mid 1930s. Artsrouni proposed a general purpose machine which could function as a mechanical multilingual dictionary and Troyanskii proposed a scheme for coding interlingual grammatical rules and provided an outline of how analysis and synthesis might work. However, until the end of the 1950s, the ideas of Troyanskii were not known that well. Invention of computers

shifted the focus of research on machine translation and their ideas were not taken further (Hutchins, 1997).

2.3 Beginning of computerized MT (1946-1954)

In 1946-47, researchers started working in the direction of using newly invented computers for translating natural languages. Within a few years, research on MT started at many US universities, and in 1952, first major project of MT was started at Georgetown University. This project was known as Georgetown Automatic Translation (GAT). In 1954, first public demonstration of the feasibility of machine translation was given by a team consisting of experts from IBM and researchers from Georgetown University. They took a sample of 49 Russian sentences and translated these to English. The demonstration attracted a great deal of media attention in the US. Massachusetts Institute of Technology (MIT) published first issue of 'Mechanical Translation' journal in the same year under the editorship of Victor Yngve. The first book on MT edited by Locke and Booth was published in 1955 (Hutchins, 2003).

2.4 Decade of high expectations and disillusion (1955-1966)

This decade saw the beginning of three basic approaches to MT. These are Direct translation approach, Interlingua approach and Transfer approach. The researchers under Erwin Reifler followed the dictionary based direct approach. They constructed large bilingual dictionaries in which lexicographic information was used not only for selecting lexical equivalents but also for solving grammatical problems without using syntactic analysis. After initial work on German and English, the group also worked on a Russian-English system (Hutchins, 1995).

During this decade, researchers at the RAND Corporation undertook statistical analysis of a large corpus of Russian Physics texts, to extract bilingual glossaries and grammatical information. On this basis, they wrote a computer program for a rough translation. The result was studied by post-editors, the glossaries and the rules were revised, the corpus was translated again, and it continued in cycles of translation and post-editing in order to translate the Russian Physics text to English. David Hays was instrumental in developing first syntactic parser based on dependency grammar at RAND Corporation (Hutchins, 1995).

The Linguistics Research Center (LRC) at the University of Texas, founded by Winfried Lehmann in 1958, concentrated on basic syntactic research. They devised reversible grammars to achieve bi-directional translation with syntactic transfer approach, laying foundations for the later development of the *'METAL'* (Mechanical Translation and Analysis of Languages) system (Hutchins, 1995).

University of Grenoble formed a MT group headed by Bernard Vauquois in year 1960. This group was first known as *'CETA'* (Centre d'Etudes pour la Traduction Automatique 'Center for the study of MT') and focused on translation of Russian text into French. They concentrated on the development of formalisms for expressing linguistic information. The system performed dependency-structure analysis of every sentence. The theoretical basis of *'CETA'* was interlingua at the grammatical level, but transfer at the lexical level. The system was written in IBM assembly language. With the use of this system it was concluded that, the use of an interlingua erases all clues about how to express the translation and it resulted into extremely poor or no translation of sentences for which complete analysis could not be derived (Slocum, 1984).

In this decade itself, *'GAT'* system became operational in 1964, with its delivery to the Atomic Energy Commission of United States. It had three levels of analysis: morphological, syntagmatic and syntactic (Hutchins, 1995). The system was primarily used to translate Russian Physics texts into English. The output quality was quite poor, and the system used to quickly scan the documents to determine their content and interest. *'GAT'* system design did not include any computational or linguistics theory and it was developed to work for a given text and then it was modified to account for the next text, and so on. As such, the result was a monolithic system of intractable complexity that caused its termination in late 1960s (Slocum, 1984).

A *'Logos'* machine translation system, developed by Logos Corporation stared in 1964. It was a direct Machine Translation system for English-Vietnamese language pair. *'Logos'* analyzed whole source sentences, considering morphology, meaning, and grammatical structure. It had easily extensible dictionaries with their underlying semantic foundation and it permitted meanings to be deduced using contextual clues. This system has kept on changing. Even the approach of translation was also changed at one point of time, from direct approach to transfer approach (Yurtseven, 1997).

University of the Saar at Saarbruecken, West Germany started a major MT project, namely, *'SUSY'* in late 1960s. In this project, a multilingual German system involving German, Russian, English and French languages was proposed (Hutchins and Somers, 1992). The translation phases of this system included morphological analysis, phrasal analysis, semantic disambiguation, transfer, semantic synthesis, syntactical synthesis and morphological synthesis. The system was written in FORTRAN (Aref *et al.*,1995).

The Soviet Union research mainly focused on interlingua approach in this decade. They considered interlingua as an artificial language, complete in itself with its own morphology and syntax and having only those features that are statistically most common to large number of languages (Hutchins, 2003).

Machine translation research faced a very difficult time during 1966 when the Automatic Language Processing Advisory Committee (ALPAC) report was published. This report concluded that machine translation was more expensive, less accurate and slower than human translation. The report also argued that despite the expenses, machine translation was not likely to reach the quality of a human translator in the near future. The report, however, recommended that tools like automatic dictionaries should be developed to aid translators and research in computational linguistics should continue to be supported. The publication of the report had a great impact on research in the area of machine translation (Hutchins, 1995).

2.5 Post ALPAC decade (1967-1976)

Focus of research in MT shifted to indirect approaches from direct approaches during this decade (Hutchins, 2003). In Georgetown University, researchers continued their efforts and proposed a machine translation system, namely, *'SYSTRAN'* in 1968. This system had four stages of translation, namely, preprocessing, analysis, transfer and synthesis (Hutchins and Somers, 1992; Aref *et al.*, 1995). This decade also achieved a landmark in the form of first doctoral thesis in MT by Anthony G. Oettinger which carried the study for a Russian mechanical dictionary in 1970 (Hutchins, 2003). The *'CETA'* group took a new project entitled *'GETA'* (Groupe d'Etudes pour la Traduction Automatique 'Group for the study of MT') for redesigning earlier system and the group came with a new system, namely, *'Ariane'* that used transfer approach (Hutchins and Somers, 1992). This system supported three pairs of languages, Russian-French, German-French, and

Japanese-French. The translation process in this system involved morphological analysis of source text, a multilevel analysis to give an intermediate source structure, lexical transfer of source-language lexical units to the corresponding target-language lexical units, a structural transfer to produce an intermediate target structure, and finally a syntactic morphological generation (Aref *et al.*, 1995).

The *'TAUM'* (Traduction Automatique de l'Université de Montréal) project was undertaken at Montreal in 1970 using syntactic transfer system for English-French translation. This system developed in Prolog was used in translating weather forecast between English and French (Hutchins, 1995).

Brigham Young University (BYU) established a project to translate church texts from English into multiple languages like French, German, Portuguese and Spanish in 1971. The main objective here was to produce a fully-automatic MT system based on Junction Grammar (Slocum, 1984).

2.6 The revival of research (1977 to 1989)

This decade witnessed the revival of MT research and a good number of MT systems were developed across the world during this decade. During 1980, Grenoble group (*'GETA'*) developed its influential *'Ariane'* system. This system is considered as 'second generation' linguistics-based transfer system. Using a similar approach, *'Mu'* system was developed at the University of Kyoto, Japan. The prominent features of *'Mu'* were: (i) use of case grammar analysis and dependency tree representations, and (ii) development of a programming environment for grammar writing (Hutchins, 2003). In 1981, a project named *'Eurota'* was started for dealing with all nine official languages of European Council. It was a transfer based system with 72 language pairs. In the proposed system, the translation was viewed as a mapping by using rules though source language analysis, transfer and generation of target language. *'Eurotra'* was implemented using Prolog on UNIX system (Aref *et al.*, 1995). Distributed Language Translation (*'DLT'*) project to develop an interlingual machine translation system for twelve European languages stared in 1985. This project used *'Esperanto'* as an Intermediate Language (IL). *'DLT'* consisted of two translation systems; source language to *'Esperanto'* and *'Esperanto'* to target language (Witkam, 1988). Uchida and Sugiyama (1980) observed that transfer approach is effective for translation among

languages of the same family and it is not that effective for translation among non-related languages such as Japanese and English due to requirement of large structural transformation. They introduced the idea of conceptual structure and utilized it in translation process. This conceptual structure used concepts in source language; these concepts were considered as universally general to the extent that they can be translated into any language. The concepts were represented by nodes and an arc was used to represent a relation between these concepts. They proposed Japanese to English MT system, in which a given Japanese text is transformed into conceptual structure, and then an English text was generated from it.

In 1984, Fujitsu proposed *'ATLAS-I'* and *'ATLAS-II'* machine translation systems. The *'ATLAS-I'* is considered as world's first commercial English-Japanese translation system. It is a syntax-based machine translation system. The *'ATLAS-II'* aimed at achieving multilingual translation. It is based on a concept structure in the form of an interlingua (Uchida, 1989a). Dorr (1987) proposed *'UNITRAN'* as an interlingual machine translation system. This system relied on principal-based descriptions of grammar rather than rule oriented descriptions.

Uchida (1989b) canvassed for the necessity of interlingua for multilingual translation system. He emphasized that interlingua can localize the development of a machine translation system for a language, since it has separate analysis and generation system. The developers of analysis and generation system only need to know the language and the interlingua. He argued that interlingua makes the common utilization of the knowledge necessary for machine translation systems, which helps to eliminate the redundancy to develop the same knowledge in different format.

Tanaka *et al.* (1989) developed a multi-lingual machine translation system using interlingua approach for five languages, namely, Chinese, Indonesian, Malaysian, Thai and Japanese. Their system had six important elements that include input system, sentence analysis system, interlingua, electronic dictionary system, sentence generation system and output system.

Another interlingua project started in Netherlands during this decade. An important feature of this project was the use of reversibility of grammars, *i.e.*, compilation of grammatical rules and transformations can work in one direction for syntactic and

semantic analysis of a language and generation of correct sentences in that language in other direction (Hutchins, 2003).

2.7 Decade of 1990 to 2000

During the initial years of this decade, translating workstations were available to support professional translators. These workstations combined multilingual word processing, OCR facilities, terminology management software and translation memories for the professional translators (Hutchins, 2003).

One witnessed a number of changes in the area of machine translation during this decade. IBM published the results of experiments on *'Candide'* system based on statistical methods in earlier years of this decade. The proposed system first align phrases, word groups and individual words of the parallel bilingual texts, and then calculates the probabilities that any one word in a sentence of one language corresponds to a word or words in the translated sentence with which it is aligned in the other language. This approach was successful even in translating the phrases from one language to other (Hutchins, 1995). IBM researchers improved the results of *'Candide'* system by proposing better statistical techniques using methods from information theory and improved probability model of the translation process (Berger *et al.*, 1994).

In this decade, one also witnessed use of Example-Based Machine Translation (EBMT). Sumita *et al.* (1990) compared EBMT with conventional Rule-Based Machine Translation (RBMT). They explored EBMT's new features like context sensitive translation, robustness and reliability factor. This decade also witnessed the emergence of research on speech translation. Integration of speech recognition, speech synthesis, and translation modules by mixing rule-based and corpus-based approaches was proposed in this decade (Hutchins, 1995).

Center for Machine Translation (CMT), Carnegie Mellon University (CMU) initiated *'KANT'* (Knowledge-based, Accurate Natural-language Translation) project. The system proposed in this project processed source text and produced a grammatical functional structure, called as F-structure. This F-structure was mapped to target language F-structure and then target language sentence were generated for these F-structures of target language (Mitamura and Nyberg, 1995).

In 1996, Institute of Advanced Studies of United Nations University, Tokyo, proposed an intermediary language, known as Universal Networking Language (UNL). Initially, it was proposed for six official languages of the United Nations and other widely spoken languages, involving 15 countries. Uchida *et al.* (1999) have illustrated general idea of the UNL and its first version specifications. They have also presented the UNL system with all its components.

2.8 Research since 2000

During this period, researches have worked on hybrid systems combining statistical and rule-based techniques. Linguistic rules were used to parse the source text and statistical (or example-based) methods were used to find proper interpretation (Siddiqui and Tiwary, 2008).

Mitamura and Nyberg (2000) have re-designed and re-implemented *'KANT'* machine translation system to *'KANTOO'*. The new system includes modules for source language analysis, target language generation, source terminology management, target terminology management and knowledge source development.

A new version of *'SYSTRAN'* has been proposed in this decade. This new version has the capacity of backtracking and it also consists of a fine-grained semantic description supporting Arabic, Farsi and Urdu lexical entries. The components in this version include morphological analyzer, statistical guesser, and a statistical based post-editor (Senellart *et al.*, 2001; Farghaly and Senellart, 2003; Schwenk *et al.*, 2009).

Brockett *et al.* (2002) have proposed a hybrid example-based English-Japanese machine translation system. It employs handcrafted broad coverage augmented phrase structure grammars for parsing. It also uses statistical and heuristic techniques to capture translation knowledge. The system has used an abstract linguistic representation layer to extract and store bilingual translation knowledge.

Lepage and Denoual (2005) implemented *'ALEPH'*, a pure example-based machine translation system to neutralize translation divergences in an elegant way. Fat (2005) has proposed Tagalog-to-Cebuano Machine Translation System (T2CMT) based on transfer approach. It uses Tagalog stemming algorithm for morphological analysis. The proposed system incorporates formal grammar for the syntax analyzer which accepts the input from Part-Of-Speech (POS) tagger and performs the transfer operation through affix and root

transfers. Navarro *et al.* (2004) have developed *'SisHiTra'* a hybrid Machine Translation system from Spanish to Catalan. The proposed system has combined the knowledge-based and corpus-based techniques.

Borra *et al.* (2007) have developed an English-to-Filipino MT system using transfer based approach. It uses Lexical Functional Grammar (LFG) as its formalism. Google Translate, introduced in 2007, is based on statistical methods. It gives a translation service for 64 languages. Google Translate has been developed by Franz Josef Och and his team.

Lagarda and Casacuberta (2008) proposed AdaBoost algorithm for improving accuracy of SMT systems. The purpose of this algorithm is to find a highly accurate rule by combining many weak or base hypotheses. The results from experiments confirm that statistical machine translation can take advantage from this technique for improving the translation quality.

Schwenk *et al.* (2009) have used SMT for building a translation system between French and English. They have used *'Moses'* decoder and a statistical post-editing system. Koehn *et al.* (2009) used JRC-Acquis corpus to build machine translation systems based on statistical approach for 462 language pairs. They have identified English as the best pivot language for multilingual machine translation system.

Scott and Barreiro (2009) have proposed *'OpenLogos'* system using pipeline architecture. Their system uses a modularized, incremental approach of source language analysis and target language synthesis. The representation language SAL (Semanto-syntactic Abstraction Language) is the key to the pipeline process. It uses indexed pattern dictionaries to store the rules.

Vashee and Gibbs (2010) have worked on customization of SMT engine. Traditional, SMT requires large corpora of aligned bilingual data. In practice, however, this data is not readily available in most languages or domains. For this scenario, they have proposed a unique process to harvest and translate *n*-grams directly. The key difference in the proposed system from the traditional approach is the collection and treatment of the *n*-grams.

2.9 MT for Indian languages

As per 2001 census, Indians use 122 languages, including 22 as official languages. The top six: Hindi, Bengali, Telugu, Marathi, Tamil, and Urdu languages are spoken by approximately 850 million people worldwide (Anthes, 2010). Only about 10% of Indian population speaks English. In the age of Internet, most of information is generated in English. Automatic translation of information available in English has immerged as an important topic of research during recent times. Department of Information Technology, MoCIT, Government of India, has initiated 'Technology Development for Indian Languages' (TDIL) project in the year 1990. The main objective of this project is to support and fund research and development efforts in the area of information processing in Indian languages including machine translation (Srikanth and Kapoor, 2001). History of machine translation in India has witnessed a number of research projects covering a good number of Indian languages. Prominent among these are the projects at IIT Kanpur, IIT Bombay, IIIT Hyderabad, University of Hyderabad, National Center for Software Technology (NCST) Mumbai (now, Center for Development of Advanced Computing (CDAC) Mumbai), CDAC Pune and Jadavpur University, Kolkata.

In the next section, developments in Indian languages machine translation systems have been presented.

2.9.1 Indian languages MT systems

Research on machine translation started in India in early eighties at IIT Kanpur. Sinha (1984) proposed Sanskrit as interlingua for translation within Indian languages. Later in 1991, the concept of a *'Pseudo-Interlingua'* is introduced to exploit the structural commonality of a group of languages. The first prototype was proposed by them for English to Tamil MT and later a comprehensive system was built for English to Hindi translation. Sinha *et al.* (1995) proposed *'AnglaBharti'*, a multi-lingual translation system from English to Indian languages. The approach used in *'AnglaBharti'* is somewhere in between transfer and interlingua approaches. This approach uses pattern directed rule-base and a Pseudo Lingua for Indian Languages (PLIL). In this system, a number of semantic tags have been used to resolve sense ambiguity in the source language. The system also used a post-editing package for final corrections.

Jain *et al.* (1995) developed *'AnuBharati'*, a template-based machine translation system from Hindi to English. *'AnuBharati'* uses a hybrid approach called as Hybrid Example Based Machine Translation (HEBMT). It is an attempt to combine the strategies of rule-based approach and example-based approach. Sinha and Jain (2003) have also proposed the modified version of *'AnglaBharti'*, known as *'AnglaHindi'*, as English to Hindi machine translation system. This system, besides using the rule-bases also uses example-base and statistics to obtain more acceptable and accurate translation for frequently encountered noun and verb phrasal. It generates approximately 90% acceptable translation in case of simple, compound and complex sentences up to a length of 20 words. It makes the use of all modules of *'AnglaBharti'* (Sinha *et al.*, 1995) and an abstracted example-base of *'AnuBharti'* (Jain *et al.*, 1995).

In 2004, *'AnglaBharti'* underwent a considerable change to address the shortcomings of the earlier architecture. The proposed system, known as *'AnglaBharti-II'*, is a hybrid system which uses Rule-Based Machine Translation (RBMT) and Example-Based Machine Translation (EBMT) techniques. In 2004 itself, *'AnuBharti'* also underwent a considerable change from its initial conceptualization to *'AnuBharti-II'*. The core of *'AnuBharti-II'* architecture is a generalized hierarchical example-base. In the absence of large parallel corpora, the example-base is augmented interactively during the development phase. In the proposed system, the input Hindi sentence is processed by shallow grammatical analyzer to convert it into a standardized form. The standardized Hindi sentence is matched with a top level standardized example-base. In case no match is found then a shallow chunker is used to fragment the input sentence into units which are then matched with a hierarchical example-base. The translated chunks are positioned by matching with sentence level example-base. *'AnuBharti-II'* also has an error-analysis module based on statistical language model (Naskar and Bandyopadhyay, 2005). Sinha (2005b) argue that RBMT has its own limitations while dealing with real-life texts and proposed the integration of Computer Assisted Tools (CAT) to existing *'AnglaBharti-II'* architecture to increase the accuracy of system.

A transfer based English-Hindi Translation System has been proposed by Gore and Patil (2002). Their translation module consists of pre-processing module, English tree generator module, a module for post-processing of English tree, generation of Hindi tree

module, post-processing module for Hindi tree and Hindi generation module. Kulkarni (2003) has given a design and architecture of *'Anusaaraka'* approach for machine translation system. It is based on the principles of Paninian Grammar (PG), and exploits the close similarity of Indian languages.

'MANTRA' (MAchiNe assisted TRAnslation tool), developed by C-DAC for translating office documents from English to Hindi, views the translation process as a mapping from one syntactic tree to another, based on Tree Adjoining Grammar (TAG). The English document is initially sent for pre-processing followed by parsing and generation. It uses auto-phrase-detection algorithms to restrict the size of various lexicons (Dwivedi and Sukhadeve, 2010).

A project on English-Kannada machine-aided translation system for translating English texts into Kannada was initiated at University of Hyderabad in 1998. The proposed system works at sentence level. It parses an input sentence using Universal Clause Structure Grammar (UCSG) parsing technology and then translates it into Kannada using the English-Kannada bilingual dictionary, Kannada morphological generator and translation rules (Naskar and Bandyopadhyay, 2005).

The English-Bangla-ANUBAD or *'EB-ANUBAD'* is a hybrid scheme based machine translation system developed jointly by Jadavpur University, Kolkata and CDAC, Kolkata. It is based on hybrid architecture of rule-based and transfer architecture. It takes a paragraph of English sentences as input sentences and produces equivalent Bangla sentences. The proposed system consists of preprocessor, morphological parser, semantic parser, an electronic lexicon associated with grammatical information, discourse processor and lexical disambiguation analyzer (Saha, 2005).

Sinha and Thakur (2005) have proposed a translation system for bi-lingual Hindi-English (Hinglish) text. Their system performs the translation in seven steps. Each word of the input Complex Code-Mixed (CCM) Hinglish sentence is processed by Hindi morphological analyzer and marks the words that are unknown in Hindi. Similarly, each word of the input CCM Hinglish sentence is fed to English morphological analyzer and marks the words that are unknown in English. The words that remain unknown in earlier processing are marked for cross morphological analysis. The words that remain unknown even at this stage are marked as proper nouns. The input CCM Hinglish sentence is

segmented into Simple Code-Mixed Hindi (SCMH) and Simple Code-Mixed English (SCME) parts. The system performs the translation of SCMH into Hindi language and transforms the SCME into pseudo English form. The pseudo English form is then translated to Hindi Language. If the target language is Hindi, then final result is obtained by simply re-composing the translations. If the target language is English, then final result is obtained by translating the Hindi text to English. This system incorporates an additional layer to the existing English to Hindi translation (*'AnglaBharti-II'*) and Hindi to English translation (*'AnuBharti-II'*) systems.

A machine translation system, namely, *'Shakti'* has been developed jointly by Carneige Mellon University, USA; IIIT Hyderabad, India and IISC Bangalore, India. This system has been designed to produce translated texts in Indian languages from English language. It uses *'Shakti Standard Format'* (SSF) that acts as a blackboard around which the complete system resolves. This system combines rule-based approach with statistical approach. The rules are mostly linguistic in nature, and the statistical approach uses linguistic information (Sangal and Sharma, 2004).

Naskar and Bandyopadhyay (2006) have proposed a Phrasal EBMT system for translating from English to Bengali. They have identified the phrases in the input through a shallow analysis and retrieve the target phrases using a phrasal example-base. The generator system combines the target language phrases by using some heuristics based on the phrase ordering rules for Bengali.

English to (Hindi, Kannada, Tamil) and Kannada to Tamil language-pairs example-based machine translation system has been proposed by Balajapally *et al.* (2006). Their system makes the use of a bilingual dictionary comprising of sentence dictionary, phrases dictionary, words dictionary and phonetic dictionary for the machine translation. These dictionaries contain parallel corpora of sentences, phrases and words and phonetic mappings of words (Dwivedi and Sukhadeve, 2010).

Ramanathan *et al.* (2006) have developed *'MaTra'*, an English-Hindi MT system for general purpose texts. It uses statistical tools and techniques for shallow parsing, word-sense disambiguation, abbreviation handling, and transliteration. A Punjabi to Hindi MT system has been proposed by Josan and Lehal (2008). It is based on direct machine translation approach and uses a number of lexical resources like a bilingual dictionary

that contains Punjabi language word, its lexical category and corresponding Hindi word; ambiguous word lexicon which contains about 1000 entries covering all the ambiguous words with their most frequent meaning; bigram table that contains Punjabi bigrams along with Hindi meaning and a trigram table. The translation engine comprises of different modules, namely, named entity recognition, repetitive construct handler, word mapping, ambiguity resolution, and transliteration.

Sinha (2009) has proposed English to Urdu translation system via Hindi. The proposed system involves use of English to Hindi MT system. The input English sentences are transformed to an intermediate form known as Pseudo Lingua for Indian Languages (PLIL) with the uses of a rule-base. The PLIL representation is then transformed to the Hindi language by using a text generator. Then English to Hindi lexical database is used to derive Hindi to Urdu mapping. The English-Hindi lexical database has syntactic, semantic and morphological information for each of the lexicon items with Hindi meaning. As Hindi and Urdu are structurally similar and use similar postpositions in the same order, the Hindi-Urdu mapping table is created to store Urdu meanings along with the information that affect the Urdu text composition. An Urdu language model is used to resolve the disambiguation in the final generation of Urdu text.

Ramanathan *et al.* (2009) have analyzed the issue of case markers and morphology to address the fluency problem in English-Hindi SMT. Their system augmented the aligned corpus of the two languages, with the correspondence of English suffixes and semantic relations with Hindi suffixes and case markers to improve the efficiency of English-Hindi SMT.

'Sampark' is a multipart machine translation system developed with the combined efforts of 11 institutions under 'Indian Language to India Language Machine Translation' (*'ILMT'*) project. The proposed system has developed language technology for 9 Indian languages resulting in MT for 18 language pairs. It uses Computational Paninian Grammar (CPG) approach for analyzing a language. It is a hybrid system consisting of traditional rule-based algorithms and dictionaries and statistical machine-learning techniques. It has three major components: source analysis which includes tokenizer, morphological analyzer, part-of-speech tagger (based on statistical techniques), chunker (based on statistical methods), named entity recognizer, simple parser based on

CPG and word sense disambiguation; transfer system which includes syntax transfer, lexical transfer and transliteration; target language generator which performs gender-number-person agreement between related words in the target sentence, insertion of *'vibhakti'* to add postposition and other markers and word generator. Their system is available online to provide the translation service among Indian languages (Anthes, 2010).

Goyal and Lehal (2010) have proposed a Hindi to Punjabi machine translation system based on direct approach. The proposed system consists of pre-processing, word-to-word translation using Hindi-Punjabi lexicon, morphological analysis, word sense disambiguation, transliteration and post processing modules.

Chaudhury *et al.* (2010) have proposed an expert system, namely, *'Anusaaraka'* that provides an access to all the stages of translation making the whole process transparent. This system has been provided by making the use of an open source MT system *'Apertium'* and an older version of *'Anusaaraka'*. Singh and Bandyopadhyay (2010) have proposed a Manipuri-English bidirectional statistical machine translation system by using factored model. They have identified suffixes and dependency relations of English and case markers of Manipuri as the important translation factors for the proposed statistical machine translation system. The proposed system has been trained using a parallel corpus from news domain.

IBM India Research Lab at New Delhi has also been working on English-Hindi MT system. Initially, their approach was based on EBMT and now the focus has been shifted to SMT for the development of MT system. Tamil-Hindi and English-Tamil translation systems have been proposed by AU-KBC Research Centre, Anna University, Chennai. These systems are based on the *'Anusaaraka'* MT system architecture. Tamil morphological analyzer and Tamil-Hindi bilingual dictionary are by products of this system (Dwivedi and Sukhadeve, 2010).

2.10 Research activities in Universal Networking Language

Research on UNL has three distinct divisions. These divisions are: development of EnConversion and DeConversion modules; applications of UNL in other contexts such as knowledge representation and knowledge management, multilingual search engines, language-independent Universal Digital Library *etc.*; and use of external lexical and

ontological resources like WordNet to enhance the processes of UNL (Cardeñosa *et al.*, 2005c). In this section, we have provided the review of literature in these major areas.

2.10.1 Development of EnConversion and DeConversion modules

There are three different approaches that have been used to design and develop the tools for EnConversion and DeConversion processes. In the first approach, one uses a common engine like *'EnCo'* and *'DeCo'* tools provided by the UNDL center to accomplish the task (UNDL Foundation, 2010). The other approach is integrative approach that is based on the integration of UNL into pre-existing MT systems. Third approach is followed by researchers who have noticed the drawbacks in the tools provided by the UNDL center, and have created new architectures from scratch. Martins *et al.* (1997) have proposed a prototype system for converting Brazilian Portuguese into the UNL and DeConverting UNL expressions into Brazilian Portuguese with *'EnCo'* and *'DeCo'* tools respectively. Their system consists of three important sub-modules, namely, the lexical, the syntactic and the semantic modules. Sérasset and Boitet (2000) have viewed UNL as the future 'html of the linguistic content'. The French DeConverter proposed by them is based on the modular architecture. It splits the DeConversion process into transfer and generation steps. Multilingual information processing through UNL has been proposed by Bhattacharyya (2001) and Dave *et al.* (2001). Their system performs sentence level encoding of English, Hindi and Marathi into the UNL form and then decodes this information into Hindi and Marathi. As such, the system has created a way of semi-automated translation from English to Hindi and Marathi and also between Hindi and Marathi. Lafourcade and Boitet (2002) have found that during DeConversion process there are some lexical items, called UWs, which are not yet connected to lemmas. They have suggested the necessity to find a nearest UW and have proposed the concept of 'conceptual vectors' for this purpose.

Martins *et al.* (2002) have analyzed unique features of UNL taking inferences from Brazilian Portuguese-UNL EnConverting task. They have suggested that UNL should not be treated as an interlingua, but as a source and a target language owing to flexibility that EnConversion process brings to UNL making this just like any other natural language.

Dhanabalan *et al.* (2002) have proposed an EnConversion tool from Tamil. Their system uses existing morphological analyzer of Tamil to obtain the morphological features of the

input sentence. They have also employed a specially designed parser in order to perform syntactic functional grouping. The whole EnConversion process has been driven by the EnConversion rules written for Tamil language.

Dhanabalan and Geetha (2003) have proposed a DeConverter for Tamil language. It is a language-independent generator that provides synchronously a framework for word selection, morphological generation, syntactic generation and natural collocation necessary to form a sentence. The proposed system involves the use of language specific, linguistic based DeConversion rules to convert the UNL structure into natural language sentences. Martins *et al.* (2005) have noted that the *'EnCo'* and Universal Parser tools provided by UNDL foundation require inputs from a human expert who is seldom available and as such their performance is not quite adequate. They have proposed the *'HERMETO'* system which converts English and Brazilian Portuguese into UNL. This system has an interface with debugging and editing facilities along with its high level syntactic and semantic grammar that make it more user friendly.

Mohanty *et al.* (2005b) have used Semantically Relatable Sequence (SRS) based approach for developing a UNL based MT system. They have analyzed the source language using semantic graphs and used these graphs to generate target language text. Dey and Bhattacharyya (2005) have presented the computational analysis of complex case structure of Bengali for a UNL based MT System. They provided the details of the rule theory of *'EnCo'* and *'DeCo'* tools which are driven by analysis rules and generation rules respectively for Bengali language. These rules are based on case structure of '*kaaraks'* used in the Bengali language. Mohanty *et al.* (2005a) have investigated the problem of prepositional phrase (PP) attachment in the context of UNL based MT systems for English language. They have performed the linguistic analysis of six common prepositions in English, namely*, 'for', 'from', 'in', 'on', 'to'* and *'with'*. The insights obtained from this analysis have been used to enrich a lexicon and a rule base, for the *'EnCo'* tool.

Blanc (2005) has performed the integration of *'Ariane-G5'* to the proposed French EnConverter and French DeConverter. *'Ariane-G5'* is a generator of MT systems. In the proposed system, EnConversion takes place in two steps; first step is analysis of the French text to produce the representation of its meaning in the form of a dependency tree

and second step is lexical and structural transfer from the dependency tree to an equivalent UNL graph. Its DeConversion process also takes place in the two steps. The first step is lexical and structural transfer from the UNL graph to an equivalent dependency tree and second step is the generation of the French sentence.

Boguslavsky *et al.* (2005) have proposed a multi-functional linguistic processor, *'ETAP-3'*, as an extension of *'ETAP'* machine translation system to a UNL based machine translation system. The proposed system is used to build a bridge between UNL and the internal representations of *'ETAP'*, namely Normalized Syntactic Structure (NormSS). The system has performed the resolution of ambiguity with the linguistic knowledge base of *'ETAP-3'*. They have also proposed an interactive system that helps to resolve difficult cases of linguistic ambiguity by means of a dialogue with the human.

Shi and Chen (2005) have proposed UNL DeConverter for Chinese language. They have highlighted the problems of *'DeCo'* tool provided by the UNDL center which includes difficulty in writing the rules, its slow speed and non-availability of the source code. These issues motivated the developers to propose a new DeConverter for Chinese. Pelizzoni and Nunes (2005) have introduced *'Manati'* DeConversion model as a UNL mediated Portuguese-Brazilian sign language human-aided machine translation system. The system proposed by them is based on constraint programming to reduce search; while object-oriented and higher-order programming provides a basis for defining friendly primitives. Daoud (2005) has proposed an Arabic DeConversion system which involves mapping of relations, lexical transfer, word ordering, and morphological generations. In the mapping of relations phase, each UNL relation has been mapped to the corresponding Arabic grammar structure which is implemented by DeConversion rules. Its word ordering phase is governed by DeConversion rules which are used during the insertion of a new node from the graph to the node-list. Arabic morphological generations is achieved by implementing a modular approach for coding the generation rules.

Keshari and Bista (2005) have proposed the architecture and design of UNL Nepali DeConverter for *'DeCo'* tool. The proposed system has two major modules, namely, syntax planning module and morphology generation module. Choudhury *et al.* (2005) have proposed a framework for converting Bangla to UNL and have also proposed a

procedure to construct Bangla to UNL dictionary. The system developed by Lafourcade (2005) uses ant colony algorithm for semantic analysis and fuzzy UNL graphs for EnConversion process. Nguyen and Ishizuka (2006) have trained UNL relations classifier using statistical techniques based feature extractor indicating the usage of statistical approach.

Singh *et al.* (2007) have proposed a DeConverter for Hindi Language known as *'HinD'*, indicating the non-availability of the source code of *'DeCo'* tool and its complex rule-format. Their system consists of four main stages including; lexeme selection, morphological generation of lexical words, function word insertion, and syntax planning. All these components use language-independent algorithms operating on language-dependent data.

Lexicalized probabilistic parser has been employed by Jain and Damani (2009) for English to UNL conversion. The parser is used to create typed dependency tree and phase structure tree for a given English sentence. Arabic MT System based on UNL has also been developed and successfully tested. In this system, Arabic generation grammar is created for the tools for MT system (Alansary *et al.* 2007; Adly and Alansary 2009). Mridha *et al.* (2010) have proposed a Bangla EnConversion system. They have analyzed Bangla words morphologically in order to obtain their roots and primary suffixes. The rules have been developed according to EnConversion specifications for generation of UNL.

2.10.2 Application of UNL in other contexts

Researchers have explored the use of UNL in other contexts such as knowledge representation, knowledge management, multilingual search engine, language-independent Universal Digital Library *etc.* In this section, we present the initiatives of researchers in this direction.

Mukerjee *et al.* (2005) used UNL as a tool for language-independent semantics for Question Answering system. This concept has further been explored by Shukla *et al.* (2004) suggesting two major tasks in the implementation of proposed Question Answering system. The first task is document processing that converts the document into equivalent UNL expression. For this, the document is processed through a UNL Parser to get the complete set of semantic predicates for the document. The other task is query

processing and answer extraction. It involves the tasks of question analysis and processing to generate the answer template; conversion of Natural Language (NL) answer template to the NL semantic predicates; conversion of NL semantic predicates to interlingua; search engine and the conversion of the answer from the interlingua to NL through the DeConversion process.

Bértoli *et al.* (2005) have proposed *'CELTA'* showcase as a web platform for using UNL. They demonstrated a multilingual business-to-business platform using UNL. Surve *et al*. (2004) have proposed an *'Agro-Explorer'* as a meaning based, interlingua search engine designed specifically for the agricultural domain covering English, Hindi and Marathi languages. The system involves the use of *'EnCo'* tool for the EnConversion of English query to UNL. The query in the UNL expression searches the UNL corpus of all documents. When a match is found, it sends the corresponding UNL file to DeConverter to provide the contents in the native language. Ramamritham *et al.* (2006) have further improved *'Agro-Explorer'* to develop *'aAQUA'* an online multilingual, multimedia agricultural portal for disseminating information from and to rural communities. The *'aAQUA'* makes use of novel database systems and information retrieval techniques like intelligent caching, offline access with intermittent synchronization, semantic-based search *etc.*

Tomokiyo and Chollet (2003) have proposed a VoiceUNL to represent speech control mechanisms within the UNL framework. The proposed system has been developed to support Speech to Speech Machine Translation (SSMT). The UNL is, however, designed to support written texts; the developers explored the possibility of extending the UNL attributes tags to suit to the SSMT processing.

Choudhary and Bhattacharyya (2002) have performed the text clustering using UNL representation. They have generated feature vectors using UNL. The clustering method used in the proposed system is the Self Organizing Maps (SOMs). They have inferred that UNL method for feature vector generation performed better than frequency based methods. Jiang *et al.* (2005) have explored UNL as a facilitator for communication between languages and cultures. They designed a system to solve critical problems emerging from current globalization trends of markets and geopolitical interdependence.

It facilitates the participations of people from various linguistic and cultural backgrounds to construct UNL knowledge bases in a distributed environment.

Cardeñosa *et al.* (2005b) have proposed an eXtended Markup Language (XML) UNL model for knowledge-based annotation. They have emphasized the need to develop new labels or annotations for the complex information extraction. They have proposed usage of UNL for semantic analysis to overcome the limitations of current text-based analysis techniques.

A *'Pivot'* XML based architecture for multilingual, multiversion documents through UNL has been proposed by Hajlaoui and Boitet (2005). Montesco and Moreira (2005) have proposed Universal Communication Language (UCL) derived from UNL. UCL has been used as the language for communication among software agents and among humans too. UCL has been defined using XML to make it easier to integrate with Internet.

Iyer and Bhattacharyya (2005) have proposed the use of semantic information to improve case retrieval in case-based reasoning systems. They have proposed a UNL based system to improve the precision of retrieval by taking into account semantic information available in words of the problem sentence. The proposed system makes use of WordNet to find semantic similarity between two concepts.

Alansary *et al.* (2006) have proposed the concept of language-independent Universal Digital Library within UNL framework. A UNL based Library Information System (LIS) has been implemented as a proof of concept. The UNL LIS provides access to the books of different languages into the native language of the user. The users can search the book catalog by entering a query in their natural language. The query is EnConverted to UNL and the search is performed. The search results contain the metadata of the books matching the query. These search results are then DeConverted by invoking the corresponding Language Server and displayed to user in his native language. Furthermore, the contents of the books are also converted to UNL and added to the UNL Encyclopedia, which contains the set of all UNL documents. The contents of these books can also be DeConverted and displayed in a natural language.

Karande (2007) has proposed a multilingual search engine with the use of UNL. The proposed system requires EnConverter to convert the contents of source language to UNL. Boitet *et al.* (2007) have built Hindi-French-English-UNL resources for *'SurviTra'*

(Survival Translator) project. It is a web service, initially being developed to help a French visitor needing to communicate with an Indian helper when English is not an option. It is a bilingual chat web service equipped with a phrasebook and a dictionary. Avetisyan and Avetisyan (2010) have proposed a *'LOOK4'* system for enhancement of web search results with Universal Words (UWs) and WordNet. They have used some existing resources such as online services provided by major search engines, semantic ontology of WordNet and flexibility of UNL for representation of semantic relations among concepts.

2.10.3 Use of external resources to enhance UNL processes

The researches have been working on the use of lexical and ontological resources for the improvement of the UNL processes. In this section, we present different activities in this direction.

Sornlertlamvanich *et al.* (2000) have performed the Thai lexical semantic annotation by UWs. The process proposed by them consists of word extraction, word sense disambiguation, and UW annotation. They have also proposed a computable method to find the appropriate UW to annotate a Thai word. Iraola (2005) has used WordNet for linking UWs of Spanish-UNL dictionary. The proposed system has enriched UWs with semantic information. Verma and Bhattacharyya (2005) have performed automatic generation of multilingual lexicon for English, Hindi and Marathi by using English, Hindi and Marathi WordNets. In addition to the WordNet, the system also makes use of word sense disambiguator, an inferencer and the knowledge base of the UNL.

Ribeiro *et al.* (2004) have proposed the development of a Portuguese-UNL dictionary using the WordNet.PT, a lexical database developed under the EuroWordNet framework. This system has explored the similarity between the WordNet.PT's lexical semantic relations and the UNL. Bueno *et al.* (2005) have worked on the systematization of knowledge acquisition process for its use in intelligent management systems. Their efforts resulted into a knowledge engineering suite to support the construction of ontologies.

Bekios *et al.* (2007) have proposed a strategy for building UW dictionary with the use of WordNet. They have proposed six rules for the creation of semantic restrictions of UWs. Each Wordnet word, *i.e.*, the set of senses for the word and its lexical relations, is used as

input to the rule base and the system yields a semantic restriction suitable for the UW that is being created.

Rouquet and Nguyen (2009) have proposed an interlingual annotation of texts. They have explored the ways to enable multimodal multilingual search in large collections of images accompanied by texts. They have used the domain ontology and UNL UWs as the interlingual lexemes for annotation. Boudhh and Bhattacharyya (2009) have proposed the unification of UW dictionaries by using WordNet ontology. They have used the WordNet ontology and proposed an extension of UW dictionary in the form of U++ UW dictionary. They have used the concept of similarity measures to recognize the semantically similar context.

Chapter Summary

In this chapter, the research activities in the area of machine translation have been documented. It has been argued that different types of MT systems are required to meet widely differing translation needs. For multilingual machine translation system, interlingua is always the first choice for researches. The development of UNL as a standardized interlingua has played a vital role in bridging the language barriers through multilingual MT systems. UNL also emerged as an important standard for the applications like knowledge representation, knowledge management, multilingual search engine, language-independent Universal Digital Library *etc.*

Chapter 3

UNL Framework and Creation of Punjabi-Universal Word Lexicon

3.1 Introduction

UNL is a language that enables computers to process information across different languages. It is used to replicate the functions of natural language used by humans for communication. It has the capability to represent information and knowledge provided by natural language. The motivation behind the development of UNL is to propose an interligual representation such that semantically equivalent sentences of all languages have the same representation.

The UNL Programme was launched in 1996 in the Institute of Advanced Studies (IAS) of United Nations University (UNU) in Tokyo, Japan. In January 2001, UNU established an autonomous organization, the Universal Networking Digital Language (UNDL) Foundation, to be responsible for the development and management of the UNL Programme. From the very beginning, researchers from all over the world have been working on creation of linguistic resources and software for UNL systems.

The EnConverter and DeConverter are the core software in a UNL system. The process of converting a source language (natural language) expression into the UNL expression is referred to as EnConversion and the process of converting UNL expressions into a target language representation is called DeConversion. These two software perform their functions on the basis of a set of grammar rules and a word dictionary of target language.

UNL Language Servers can be connected to the Internet to carry out the conversions between natural languages and UNL expressions. Each Language Server will thus contain an EnConverter and a DeConverter of a language as shown in Figure 3.1.

Figure 3.1: A UNL system

3.2 UNL format for information representation

UNL represents the information in three different types of semantic units, namely, Universal Words (UWs), Relations and Attributes. UNL represents information sentence by sentence (Uchida and Zhu, 2001). Each sentence is converted into a hyper-graph (also known as UNL graph) having concepts represented as nodes and relations as directed arcs. The concepts are represented by UWs and UNL relations are used to specify the role of each word in a sentence. The subjective meanings intended by the author are expressed through UNL attributes. UNDL has formally defined the specifications of UNL (Uchida, 2005).

3.2.1 Universal Words

Universal Words form the vocabulary of UNL. These words correspond to the nodes that are interlinked by relations or modified by attributes in a UNL graph. The concepts of UWs are divided into four categories, namely, nominal concepts, verbal concepts, adjective concepts and adverbial concepts. A UW is a character string (an English language word) followed by a list of constraints (Uchida, 2005). The syntax used for describing a UW is:

<UW> ::= <headword> [<constraint list>] …(3.1)

Here, the headword is an English word or a phrase or a sentence that is interpreted as a label for a set of concepts. This is also called a basic UW. The constraint list is used to describe the sense of word. This list contains the UNL relations that represent the objective knowledge of concept in headword. The constraint list is also used to restrict the range of the concepts that an English expression represents and thus also helps in disambiguation.

The UWs are divided into four types: Basic UWs; Restricted UWs; Extra UWs and Temporary UWs. Basic UWs are bare headwords with no constraint list, for example, *'water'*, *'go'*, '*run' etc.* A UW with constraint list is known as Restricted UW. For example, the basic UW '*run'*, with no constraint list, denotes the concepts of 'management of an organization by a person' (*e.g.*, 'He is running the organization very well'), 'driving of vehicle' (*e.g.*, 'He runs the car very fast'), 'move at a speed faster than walking' (*e.g.*, 'He runs very fast in the race'), 'accumulation of bill' (*e.g.*, 'Your bill runs from January 1, 2011 to December 31, 2011'), 'movement of finger on keyboard' (*e.g.*, 'She ran her fingers on the keyboard'), 'taking risk by a person' (*e.g.*, 'Owing to his habitual lateness he ran the danger of being fired'), 'short visit' (*e.g.*, 'He runs up to Mumbai'), and so on. The UW '*run'* is restricted with constraint list to represent these concepts unambiguously as shown below:

'run(agt>human,equ>manage,obj>organization)', '*run(agt>human,equ>drive, obj> vehicle)'*, '*run(agt>human)'*, *'run(agt>human,equ>accumulate,obj>bill)'*, *'run(agt> human,opl>keyboard,obj>finger)'*, *'run(agt>human,obj>risk)'*, *'run(agt>human,pur> short visit)'*.

Here, *'agt'*, *'equ'*, *'obj'*, *'opl'* and *'pur'* are UNL relations. Extra UWs are special type of restricted UWs that are used to represent the concepts that are not available in English language. These words are represented as headwords using English characters and constraint list gives idea about the nature of the concept. For example, '*vaisakhi(icl>festival)'*, '*bhangra(icl>dance)'*, '*saag(icl>food)' etc.* Temporary UWs may not have a definition. For example, a number or an e-mail or URL need not be necessarily defined. These can appear in a UNL document and are treated as temporary UWs. The words from other languages are linked to these UWs and are assigned syntactic and semantic attributes. This forms the core of the lexicon building activity.

In the UNL specifications proposed in 2010, a simple UW is considered as an integer that can also be represented as a unique character string for better readability. This character string is split into two different parts: a root and a suffix. The root can be a word, an expression, a phrase or even an entire sentence in a language which can be interpreted as a label for a concept. The suffix is used to disambiguate the root. The UWs are considered equivalent to the sets of synonyms of a given language, similar to the concept

of synset in a WordNet. The UWs have been proposed in such a manner that language groups can create their own language specific UNL dictionaries automatically with the use of their corresponding language WordNet. The UWs are organized in a hierarchy (the UNL ontology), which are defined in the UNL knowledge-base and exemplified in the UNL example-base (lexical databases for UNL) provided by UNDL center (UNDL Foundation, 2010).

3.2.2 UNL relations

Relations are basic building blocks of UNL sentences. UNL expressions are formed using binary relations and a binary relation includes a UNL relation and two UWs. The relations between UWs have different labels according to the different roles they play. There are a number of factors that should be considered in choosing a relation (Uchida and Zhu 1993; Uchida, 2005). A UNL binary relation is of the form *rel (arg1, arg2)*. Here, *rel* represents the label of UNL relation and its first argument *arg1* is UW1 and second argument *arg2* is UW2. The UW1 acts as the parent of the relation whereas UW2 acts as the child of the relation. There are 46 relations in all that are defined for UNL (Uchida, 2005). The description of these relations is given in Table 3.1 (UNDL Foundation, 2010).

Table 3.1: Description of UNL relations

UNL relation	Description	Constituent elements	Examples
1. agt	defines a thing that initiates an action	Agent, Unergative verbs (intransitive verb semantically have an agent subject) like ‘sleep’, ‘snore’, ‘cough’, ‘run’, etc.	John slept … agt(slept, John) John killed Mary. agt(killed, John) … arrival of John … agt(arrival, John) …play by Shakespeare agt(play, Shakespeare)
2. and	defines a partner to have	Conjunction	… easily and quickly … and(quickly, easily) … singing and dancing …

	conjunctive relation		and(dance(agt>person), sing(agt>person))
3. aoj	defines a thing that is in a state or has an attribute	Stative verbs like 'believe', 'understand', 'know', 'have', 'possess', 'dislike', 'love', 'like', 'contain', 'include', 'involve', etc.	John believes in Mary. aoj(believes, John) John knows Mary. aoj(knows, John) John loves Mary. aoj(loves, John)
		aoj (general attribute)	John is sad. aoj(sad, John) John looks sad …. aoj(sad, John)
4. bas	defines a thing used as the basis (standard) of comparison	Basis	... more than seven. bas(more(aoj>thing,bas>thing), 7) … more than Jack. bas(more(icl>how,bas>thing), Jack(iof>person))
5. ben	defines an indirectly related beneficiary or victim of an event or state	Beneficiary	To give one's life for one's country. ben(give(agt>thing, gol>thing,obj>thing), country(icl>region)) It is good for John to … ben(good(aoj>thing), John(iof>person))
6. cag	Defines a thing not in focus that initiates an implicit event	Co-agent	… to walk with John. cag(walk(agt>volitional thing), John(iof>person)) To live with … aunt. cag(live(agt>volitional thing),

	that is done in parallel		aunt(icl>person))
7. cao	defines a thing not in focus that is in a parallel state	Co-thing with attribute	... be with you … cao(exist(aoj>thing), you)
8. cnt	defines the content of a concept	Content	The Internet: an amalgamation … cnt(Internet(icl>communication network), amalgamation(icl>harmony)) A language generator “deconverter”… cnt(language generator, deconverter.@double_quote)
9. cob	defines a thing that is directly affected by an implicit event done in parallel or an implicit state in parallel	Affected co-thing	… dead with Mary. cob(die(obj>living thing), Mary(iof>person))
10. con	defines a non-focused event or state that conditions a focused event	Condition	If you are tired, we will go straight home. aoj:01(tired(aoj>thing), you) con(go(icl>move(agt>thing, gol>place,src>place)), :01)

	or state		
11. coo	defines a co-occurrent event or state for a focused event or state	While	… worked while … talked coo(worked, talked)
12. dur	defines a period of time during which an event occurs or a state exists	During	John worked during … dur(worked, meeting)
13. equ	defines an equivalent concept	Equivalent	The deconverter (a language generator) … equ(deconverter, language generator.@parenthesis)
14. fmt	defines a range between two things	Range/from-to	… from a to z. fmt(z(icl>letter), a(icl>letter)) … from Osaka to New York. fmt(New York(iof>city), Osaka(iof>city))
15. frm	defines an initial state of a thing or a thing initially associated with the focused thing	Origin	A visitor from Japan … frm(visitor(icl>person), Japan(iof>country))
16. gol	defines a final state of object or a thing	Final state of verbs of change like ‘give’, ‘send’, etc.	… gave … to Mary. gol(gave, Mary) … sent … to Mary.

	finally associated with the object of an event		gol(sent, Mary)
17. icl	defines an upper concept or a more general concept	Included/a kind of	A bird is a (kind of) animal. icl(bird(icl>animal), animal(icl>living thing))
18. ins	defines an instrument to carry out an event	Instrument	… look at stars through a telescope. ins(look(agt>thing,obj>thing), telescope(icl>optical instrument)) … write with a pencil. ins(write(agt>thing,obj>thing), pencil(icl>stationery))
19. int	defines all common instances to have with a partner concept	Intersection	…. an intersection of tableware and cookware … int(tableware(icl>tool), cookware(icl>tool))
20. iof	defines a class concept that an instance belongs to	An instance of	Tokyo is a city in Japan. iof(Tokyo(iof>city), city in Japan)
21. man	defines a way to carry out	Manner	… move quickly. man(move(agt<thing,

	an event or the characteristics of a state		gol>place,src>place), quickly) … often visit ... man(visit(agt>thing, obj>thing), often)
22. met	defines a means to carry out an event	Method/means	… solve … with dynamics. met(solve(icl>resolve (agt>thing,obj>thing)), dynamics(icl>science)) … separate … by cutting … met(separate(agt>thing, obj>thing,src>thing), cut(agt>thing,obj>thing, opl>thing))
23. mod	defines a thing that restricts a focused thing	Modification	The whole story … mod(story(icl>tale), whole(mod<thing)) A master plan …. mod(plan(icl>idea), master(mod<thing))
24. nam	defines a name of a thing	Name	... son “Hikari” nam(son(icl>relative), Hikari)
25. obj	defines a thing in focus that is directly affected by an event or state	Unaccusative verbs like ‘die’, ‘fall’, ‘melt’, etc.	John died. obj(died, John) The snow melts. obj(melts,snow)
		obj (direct object)	John killed Mary. obj(killed, Mary) John knows Mary. obj(knows, Mary) John loves Mary.

			obj(loves, Mary)
		obj (indirect object of monotransitive verbs like ‘depend’, ‘believe’, ‘laugh’, etc.)	… depends on Mary. obj(depends, Mary) … believes in Mary. obj(believes, Mary)
		object	… construction of the building ... obj(construction, building) … interest in Physics … obj(interest, Physics) … visit to London. obj(visit, London)
26. opl	defines a place in focus affected by an event	Affected place	… pat … on shoulder …. opl(pat(icl>touch(agt>thing, obj>thing,opl>thing)), shoulder(pof>trunk)) … cut … in middle … opl(cut(agt>thing,obj>thing, opl>thing), middle(icl>place))
27. or	defines a partner to have disjunctive relation to	Disjunction	… stay or leave … or(leave(agt>thing,obj>place), stay(icl>remain(agt>thing))) Is it red or blue? or(blue(icl>color), red(icl>color))
28. per	defines a basis or unit of proportion, rate or distribution	Proportion/rate/distribution	… hours a day. per(hour(icl>period), day(icl>period))
29. plc	defines a	Place	Made in India.

	place where an event occurs, or a state is true, or a thing exists		plc(made, India)
30. plf	The place where an event begins or a state becomes true	Initial place of verbs of motion like ‘go’, ‘travel’, ‘walk’, ‘come’, etc.	John came from NY. plf(came, NY)
31. plt	defines a place where an event ends or a state that becomes false	Final place	… to travel to Patiala. plt(travel(agt>volitional thing), Patiala(iof>city))
32. pof	indicate a concept of which a focused thing is a part	Part of	The preamble of a document … pof(preamble(icl>information), document(icl>information)) … the initials of Machine Translation … pof(initial(icl>letter), machine translation)
33. pos	defines the possessor of a thing	Possessor	John’s dog. pos(dog(icl>aminal), John(iof>person)) My book. pos(book(icl>document), I)
34. ptn	defines an indispensable	Partner	… compete with John … ptn(compete(agt>thing,

	non-focused initiator of an action		ptn>thing), John(iof>person)) … share … with the poor. ptn(share(icl>divide(agt>thing, obj>thing)), poor(icl>person))
35. pur	defines the purpose or objective of an agent of an event or the purpose of a thing that exists	Purpose	*John works for money.* *agt(work(icl>do), John(iof>person))* *pur(work(icl>do), money)* … budget for research … pur(budget(icl>expense), research(icl>study))
36. qua	defines the quantity of a thing or unit	Quantity	Two cups of coffee … qua(cup(icl>tableware), 2)) qua(coffee(icl>beverage), cup(icl>tableware)) … many kilograms … qua(kilogram(icl>unit), many(qua<thing)) … two dogs … qua(dog(icl>animal), 2)
37. rsn	defines a reason why an event or a state happens	Reason	*He goes … because of … illness.* rsn(go(icl>do), illness(icl>thing)) … known for … beauty. rsn(known(aoj>thing), beauty(icl>abstract thing))
38. scn	defines a scene where an event	Scene	*… appear on a program.* scn(appear(icl>occur), program(icl>thing))

	occurs, or state is true, or a thing exists		
39. seq	defines a prior event or state of a focused event or state	After	John worked and left … seq(left, worked)
40. shd	defines a number, a mark or a thing that shows the position of a sentence, a paragraph or a chapter in a document or a book	Sentence head	Chapter 2 Relation shd(relation(icl>sate), chapter(pof>book)) mod(chapter(pof>book), 2)
41. src	defines the initial state of an object or thing initially associated with the object of an event	Initial state of verbs of change like ‘take’, ‘retrieve’, etc.	*… changed from red …* src(change(icl>occur), red(aoj>thing))
42. tim	defines the time an event occurs or a	When	… came yesterday … tim(came, yesterday)

	state is true		
43. tmf	defines the time an event starts or a state becomes true	Since when	… worked since early … tmf(worked, early)
44. tmt	defines the time an event ends or a state becomes false	Until when	… worked until late … tmt(worked, late)
45. to	defines a final state of a thing or a final thing (destination) associated with the focused thing	Destination	*… train for London …* to(train(icl>thing), London(icl>city))
46. via	defines an intermediate place or state of an event	An intermediate place or state	*… goes to … via New York.* via(go(icl>do), New York(icl>place))

3.2.3 UNL attributes

Attributes are used to describe subjective information in a sentence. These attributes exhibit the point of view of speaker in a given sentence (Uchida and Zhu 1993). There are 87 attributes (which can be augmented with the user defined ones) to express the semantic content of a sentence. UNL attributes are divided into eight groups.

Descriptions of these groups with corresponding UNL attributes are given in Table 3.2 (Uchida, 2005).

Table 3.2: Description of UNL attributes

Concept	Attributed as
Logicality of UW	@transitive, @symmetric, @identifiable, @disjointed
Time with respect to the speaker	@past, @present, @future
Speaker's view on aspects of event	@begin, @complete, @continue, @custom, @end, @experience, @progress, @repeat, @state @just, @soon, @yet
Speaker's view of reference to concepts	@generic, @def, @indef, @not, @ordinal
Speaker's emphasis, focus and topic	@contrast, @emphasis, @entry, @qfocus, @theme, @title, @topic
Speaker's attitudes	@affirmative, @confirmation, @exclamation, @humility, @imperative, @interrogative, @invitation, @polite, @request, @respect, @vocative
Speaker's feelings and judgments	Attributes to represent ability: @ability Attributes to represent beneficially: @get-benefit, @give-benefit Attributes to represent conclusion: @conclusion, @consequence Attributes to represent condition: @sufficient Attributes to represent consent/dissent: @consent, @dissent, @grant, @grant-not Attributes to represent expectation: @although, @discontented, @expectation, @wish Attributes to represent intention: @insistence, @intention, @want, @will, @need, @obligation, @obligation-not, @should, @unavoidable Attributes to represent possibility: @certain, @inevitable, @may, @possible, @probable, @rare,

	@unreal Attributes to represent emotion: @admire, @blame, @contempt, @regret, @surprised, @troublesome
Convention description	@passive, @pl, @angle_bracket, @brace, @double_parenthesis, @double_quote, @parenthesis, @single_quote, @square_bracket

Attributes are thus used to include information about time and aspect of the event, modality of the predication, reference of the entities mentioned, number and/or gender, *etc.* For example, in the sentence, 'The boy eats potatoes in the kitchen', attributes are needed to express plurality in the object ('*potato', i.e., '@pl'*), to indicate definite reference to agent ('*boy', i.e., '.@def'*), to indicate definite reference to place ('*kitchen', i.e., '.@def'*) and to denote the head of expression ('*eat', i.e., '.@entry'*) (Cardeñosa *et al.*, 2005a).

3.3 UNL sentence

UNL sentence is the basic unit of representation inside the UNL framework. It consists of nodes (UWs) interlinked with binary semantic relations and modified by attributes. There are two different ways of representing UNL sentences: the table format and the list format. In the table format, UWs and relations constitute a single structure, while in list format, UWs and relations are represented separately (Uchida, 2005). The UNL sentences are commonly represented in the table format and this format has been used in the present work. The syntax of table format is given in Table 3.3.

Table 3.3: Syntax of UNL sentence in table format

<UNL sentence>	::=	<list of relations>
<list of relations>	::=	<binary relation>[<binary relation>...]
<binary relation>	::=	<relation>[":"<Scope-ID>] "(" <source node>, <target node>")"
<source node>	::=	<UW+attributes>
<target node>	::=	<UW+attributes>
<UW+attributes>	::=	<UW>{":"<Scope-ID}[<attribute list>] ":" <UW-ID>

Here, '<' and '>' indicate a non-terminal symbol; '{' and '}' indicate a range; '[' and ']' indicate an omissible part; '...' indicates more than 0 times repetition of the front part; '::=' indicates that left part can be replaced by the right part and predefined delimiters are enclosed within " ".

The syntax of list format is given in Table 3.4.

Table 3.4: Syntax of UNL sentence in list format

<UNL sentence>	::=	"[W]" <list of UWs> "[/W]" ["[R]" <list of relations> "[/R]"]
<list of UWs>	::=	<UW+attributes> [<UW+attributes>...]
<UW+attributes>	::=	<UW>{":"<Scope-ID>}[<attribute list>]":"<UW-ID>
<list of relations>	::=	<binary relation>[<binary relation>...]
<binary relation>	::=	<source node><relation[":"<Scope-ID>]<target node>
<source node>	::=	<UW-ID>
<target node>	::=	<UW-ID>

These formats are illustrated below with the help of an example sentence.

Example Sentence: The boy eats potatoes in the kitchen. …(3.2)

The UNL sentence in table format of the sentence given in (3.2) is:

{unl}

agt(eat(icl>do).@entry, boy(icl>male person).@def)

obj(eat(icl>do).@entry, potato(icl>food).@pl)

plc(eat(icl>do).@entry, kitchen(icl>facilities).@def)

{/unl} …(3.3)

Here, *'agt'*, *'obj'* and *'plc'* are the relation labels, *'eat(icl>do)'*, *'boy(icl>male person)'*, *'potato(icl>food)'* and *'kitchen(icl>facilities)'* are the UWs, *'@entry'* is used to indicate head of whole expression, *'@def'* is used to indicate definite reference to agent and place and *'@pl'* is used to indicate plurality of object.

The UNL sentence in list format form for the example sentence given in (3.2) is:

{unl}

[W]

boy(icl>male person).@def :01

eat(icl>do).@entry :02

potato(icl>food).@pl :03

kitchen(icl>facilities).@def :04

[/W]

[R]

02agt01

02obj03

02plc04

[/R]

{/unl} …(3.4)

The list format of UNL sentence representation given in (3.4), suggests that UWs are sequenced with UW-ID from '01' to '04' under [W] and [/W] tags. The list of relations in the form of source node UW-ID, relation, and target node UW-ID is listed under [R] and [/R] tags (Uchida, 2005). As indicated earlier, a UNL sentence can also be represented in the form of a graph called as UNL graph. The UNL graph of the example sentence given in (3.2) is depicted in Figure 3.2.

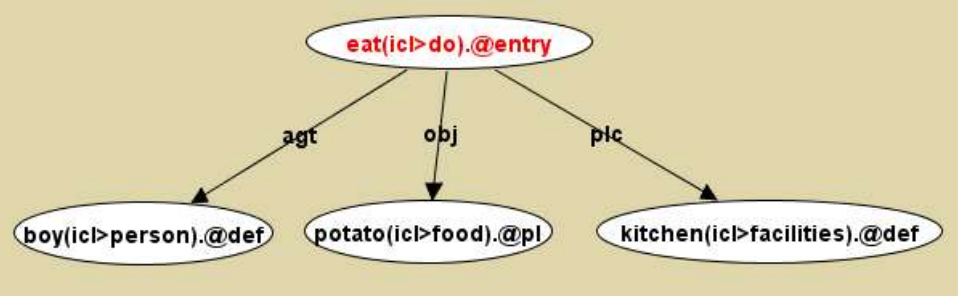

Figure 3.2: UNL graph for example sentence given in (3.2)

3.4 Compound UWs in UNL

Compound UWs are a set of binary relations that are grouped together to express a compound concept. Complex sentences are represented in UNL with the help of compound UWs. Compound UWs denote compound concepts that are to be interpreted as a whole so that one can talk about their parts all at the same time. A compound UW is expressed by a scope in UNL expressions. A Compound UW is defined by placing a compound UW-ID immediately after the UNL relation label in all of the binary relations that are to be grouped together (Uchida, 2005). Compound UW-ID starts with ':'

followed by two numbers (each between 0 and 9), for example ‘:01’. The concept of compound universal words is illustrated below with the help of one example sentence.

Example Sentence: Ram, who lives in Delhi, eats rice. …(3.5)

One can see that this is a clausal sentence involving one main clause ‘[Subject] eats rice’. Here, Subject is a complete sentence in itself represented by ‘Ram, who lives in Delhi’ acting as sub-clause of the main sentence. These, sub-clauses are represented by the compound UW as shown in following UNL expression.

{unl}

agt(eat(agt>person,obj>food), :01)

obj(eat(agt>person,obj>food), rice(icl>food))

agt:01(live(agt>person), Ram(icl>person))

plc:01(live(agt>person), Delhi)

{/unl} …(3.6)

Here, ‘:01’ indicates the compound UW. Figure 3.3, indicates the UNL graph of UNL expression (3.6).

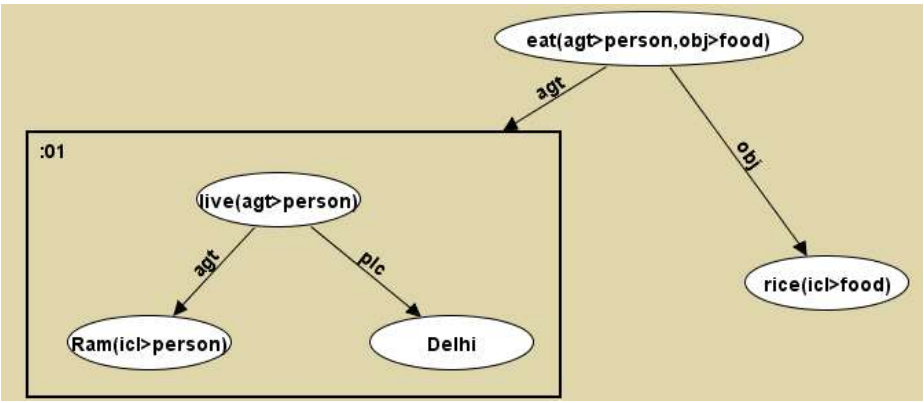

Figure 3.3: UNL graph with compound UW

3.5 UNL document

UNL documents are plain text files that contain UNL sentences and some special tags. These are the outputs produced by the process called as UNLization (EnConversion) and are taken as inputs to the process called as NLization (DeConversion). UNL document is enclosed within tags “[D:<id>]” and “[/D]”. Within these tags, each paragraph is enclosed within a pair of tags “[P:<id>]” and “[/P]”, and each sentence is enclosed within a pair of tags “[S:<id>]” and “[/S]”. The text of original sentence is enclosed within “{org:<lang>}” and “{/org}” inside sentence tags and its UNL expression is enclosed

with "{unl:<id>}" and "{/unl}". The sentences of target languages can also be stored in a UNL document. The target sentence is enclosed within a pair of language tags "{<lang>}" and "{</lang>}" following the UNL expression of each sentence (UNDL Foundation, 2010). The details of tags used in UNL document are given in Table 3.5.

Table 3.5: Tags used in a UNL document

Tag	Description
[D:<id>]	indicates the beginning of a document
[/D]	indicates the end of a document
[P:<id>]	indicates the beginning of a paragraph
[/P]	indicates the end of a paragraph
[S:<id>]	indicates the beginning of a sentence
[/S]	indicates the end of a sentence
{org:<lang>=<code>}	indicates the beginning of an original/source sentence
{/org}	indicates the end of an original sentence
{unl:<id>}	indicates the beginning of the UNL expressions of a sentence
{/unl}	indicates the end of the UNL expressions of a sentence
{<lang>}	indicates the beginning of a target sentence of the language indicated by <lang>
{/<lang>}	indicates the end of a target sentence of the language indicated by <lang>

Here, ":<id>" is optional, which is normally represented by an integer or by a sequence of characters used to identify the document, the sentence, the paragraph or the UNL expression; "<lang>" and "=<code>" tags are also optional with the {org} tag.

3.6 UNL knowledge-base

UNL Knowledge-Base (UNLKB) is used to define every possible relation between concepts, *i.e.*, UWs. The format of knowledge-base entries is given in (3.7).

<knowledge-base entry> ::= <binary relations> "=" <degree of certainty>
<binary relation> ::= <relation label> "(" <UW1> "," <UW2> ")"
<degree of certainty> ::= "0" | "1" | ... | "255" …(3.7)

Here, "::=" implies that left part can be replaced by the right part and "0" degree of certainty implies that the relation between two UWs is false and degree of certainty equal to or more than "1" implies that the relation between two UWs is true, and larger numbers represent higher credibility.

Following this scheme, the knowledge-base of sentence given in (3.8) will be represented in UNLKB as illustrated in (3.9).

A boy can run. …(3.8)

icl(boy(icl>male person),human(icl>living thing))=1

agt(run(agt>person),human(icl>living thing))=1 …(3.9)

UNLKB makes the use of three main categories of relations, namely, *'icl'*(sub class of); *'iof'* (element/instance of); and *'equ'* (equivalent to). With the use of these three relations, UNLKB is arranged hierarchically. This hierarchical structure allows implementation of the principle of inheritance in the definition of concepts. All information assigned to a parent node can be subsumed as inherited by the children nodes. For instance, in defining *'cat(icl>feline)'*, there is no need to repeat all properties of felines, as they are defined under all mammals of the same species (Nagi and Senta, 2008).

3.7 UNL system

A UNL system consists of three major components, namely, language resources, software for processing language resources, and supporting tools for maintaining and operating language processing software. The language resources are divided into language-independent parts and language-dependent parts. The language-independent components like knowledge about concepts and relations between concepts of words are stored in a common database known as UNLKB or UNL ontology. The language-dependent resources like word dictionaries, analysis and generation rules, as well as the software for language processing are stored in each Language Server (LS) connected to Internet. The supporting tools for producing language resources like UNL ontology are maintained on LS managed by UNL center. All these components help to produce UNL expressions or UNL documents from natural languages and provide a user with access to the UNL documents. Figure 3.4 depicts the process of conversion between a natural language and a UNL document when carried out in a UNL system (UNDL Foundation, 2010). The

solid arrow lines show dataflow and broken arrow lines show access. UNL system uses EnConverter and DeConverter as its core software components. The EnConverter converts natural language sentences into UNL expressions. A Universal Parser (UP) is a specialized version of the EnConverter, which generates UNL expressions from annotated sentences with referring to the UW dictionary without using grammatical features. All UNL expressions are verified by the UNL verifier, and then stored in the format of UNL document. The DeConverter converts UNL expressions to natural language sentences. Both the EnConverter and DeConverter perform their functions based on a set of grammar rules and a word dictionary of target language. The usage of UNL ontology and co-occurrence dictionary is optional in EnConverter and DeConverter tools.

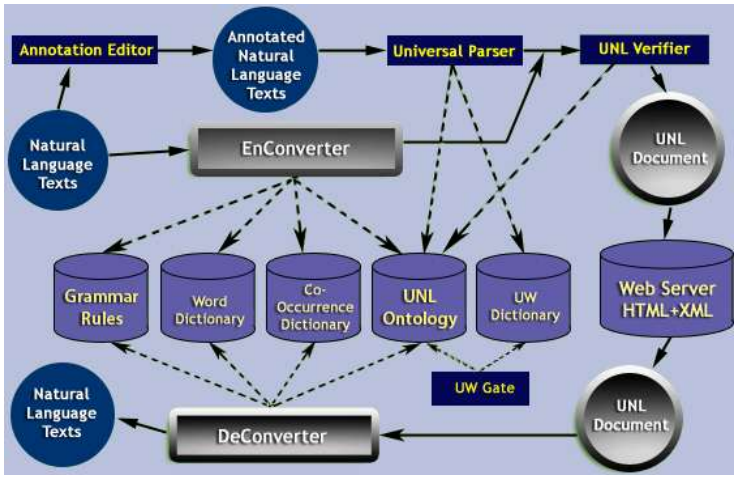

Figure 3.4: Conversion process in a UNL system

Figure 3.5 depicts the structure of a UNL system and also explains its connections with supporting tools and UNL based applications (UNDL Foundation, 2010).

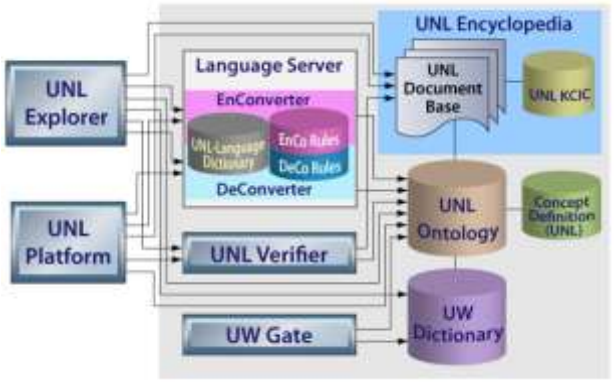

Figure 3.5: Structure of the UNL system and applications

A brief description of these components is given in the subsequent sections.

3.7.1 EnConverter

The EnConverter is a framework for morphological, syntactic, and semantic analysis of a natural language. It is a language-independent parser and uses word dictionary and set of EnConversion rules during analysis of sentences. It can deal with different natural languages by using respective languages word dictionaries and set of EnConversion rules. It takes a sentence in the form of string of text as its input and scans it from left to right. The matched morphemes from the beginning (left) of the string are retrieved from the word dictionary and become the candidate morphemes. These candidate morphemes are sorted according to priority. The EnConversion analysis rules are fired on candidate morphemes for the word selection. The syntactic tree and a semantic network for the input sentence are prepared by syntactic and semantic analysis carried out by EnConversion rules. This process continues until all words of the sentence are processed, and a complete semantic network of the input sentence is made. At the end of whole processing, the semantic network is expressed in the UNL format.

The EnConverter can also consult the UNL ontology to select appropriate UWs for ambiguous words and appropriate relations between UWs. Figure 3.6 contains the structure of the EnConverter (UNDL Foundation, 2010). Here, *'A'* indicates analysis windows, and *'C'* indicates condition windows. The EnConverter operates on the node-list through analysis windows. Condition windows are used to check conditions for applying the rule. During the initial phase of EnConversion process, only one node of an input sentence exists in the node-list while during the last phase, a syntactic tree together with a semantic network is created, and only the root node remains in the node-list. This approach of EnConversion has been used in *'EnCo'* tool proposed by UNDL foundation (UNDL Foundation, 2010).

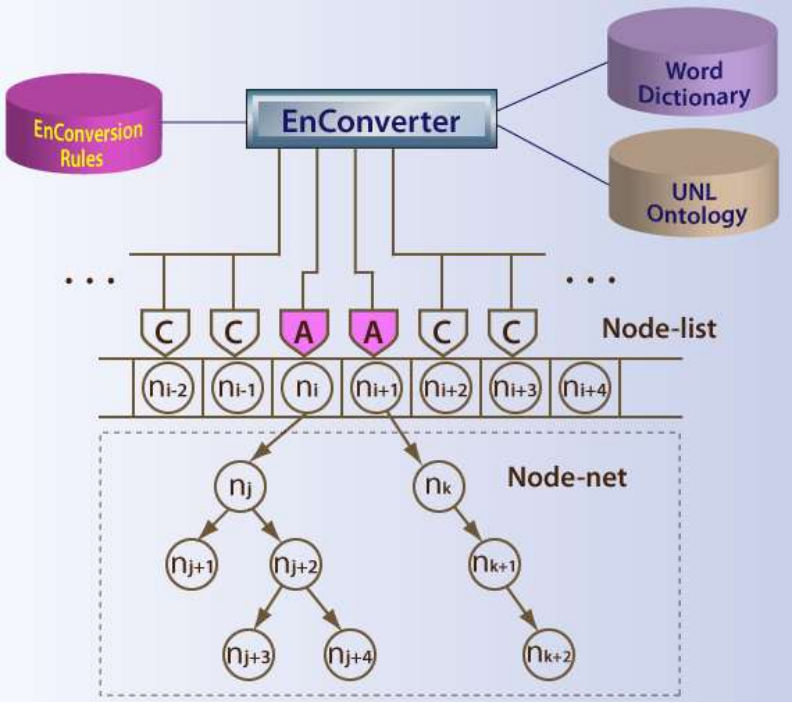

Figure 3.6: Structure of EnConverter

3.7.2 DeConverter

The DeConverter is a framework for syntactic and morphological generation from UNL to a natural language. It is a language-independent generator and can convert UNL expressions into a variety of natural languages, by using respective language word dictionaries and sets of grammar rules of DeConversion. A word dictionary contains the information of words that correspond to UWs included in the input of UNL expressions and grammatical attributes that describe the behavior of the words. The DeConversion rules describe the way to construct a sentence using the information from the input of UNL expressions and word dictionary. A DeConverter can also make use of co-occurrence dictionary of the target language for relation-based word selection for natural collocation. It can also make use of UNL ontology, when no corresponding word for a particular UW exists in a language. In this case, the DeConverter consults UNL ontology and tries to find a more general UW.

The DeConverter transforms the input, a UNL expression, into a directed graph structure with hyper-nodes. DeConversion of a UNL expression is carried out by applying DeConversion rules to these nodes. It processes the nodes from the entry node, to find an appropriate word for each node and generates a word sequence of target language. In this

process, the system makes use of syntactic rules in order to produce syntactic structure of the target language and morphological rules to generate its morphemes. The DeConversion process ends when all words for all nodes are found and a word sequence of target sentence is completed. Figure 3.7 depicts the structure of the DeConverter (UNDL Foundation, 2010). Here, *'G'* indicates generation windows, and *'C'* indicates condition windows of the DeConverter. The DeConverter operates on the node-list through generation windows. Condition windows are used to check conditions for applying a rule. In the initial stage, the entry node of a UNL expression exists in the node-list and at the end of DeConversion, the node-list is converted to list of equivalent morphemes of the target language arranged syntactically to produce target language sentence. This approach of DeConversion has been used in '*DeCo*' tool proposed by UNDL foundation (UNDL Foundation, 2010).

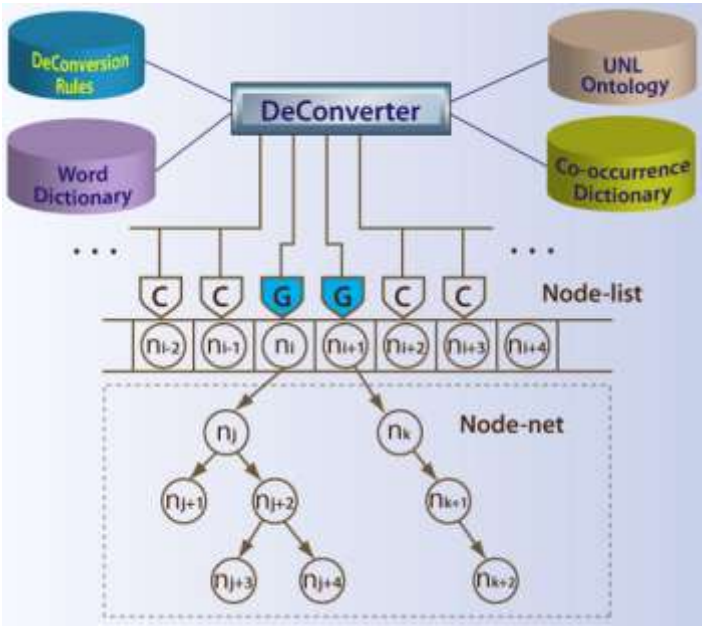

Figure 3.7: Structure of DeConverter

3.7.3 Dictionary Builder

Dictionary Builder (DicBld) is the tool provided by UNDL foundation for converting text data of dictionary entries into Index Based Access Method (IBAM) formatted dictionary files as shown in Figure 3.8 (UNDL Foundation, 2010). The UNL system makes the use IBAM to decrease the searching time of dictionaries.

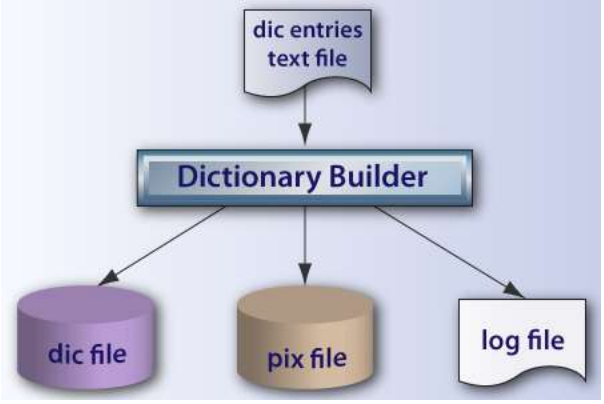

Figure 3.8: Structure of *'DicBld'*

3.7.4 Grammar rules

UNL system makes use of EnConversion and DeConversion rules of respective languages during the process of EnConversion and DeConversion. The rules are used to define actions taken by the system on some predefined conditions based on grammatical features. The rule applications can be controlled by priorities and allow backtracking. The rules can force the process of backtracking, when the EnConverter or the DeConverter encounters an ungrammatical or illogical situation. It means the system can returns back to the previous state that allows selecting a different word or morpheme, or applying the next priority rule (UNDL Foundation, 2010).

3.7.5 UNL Key Concept in Context (KCIC)

UNL uses the concept of KCIC to link every UW of the UNL ontology to the UNL documents where the UW is included. Every UW must be registered in the UNL ontology for realizing this inter-linkage of UWs crossing UNL documents (Uchida and Zhu, 2005).

3.7.6 UW Gate

The UW Gate is used to access the UNL ontology and the UW dictionary through the Internet. With this feature, the users can search for desired UWs, relations between UWs and equivalent words of desired natural languages (Uchida and Zhu, 2005).

3.7.7 Universal Parser

The Universal Parser (UP) generates UNL expressions from sentences by using language-independent annotations. It does not use language-dependent grammatical information. The sentences inputted to UP must be annotated with UNL annotation

(Uchida and Zhu, 2003). The UP analyzes the annotated input sentences using UP rules and a UW dictionary as shown in Figure 3.9 (UNDL Foundation, 2010).

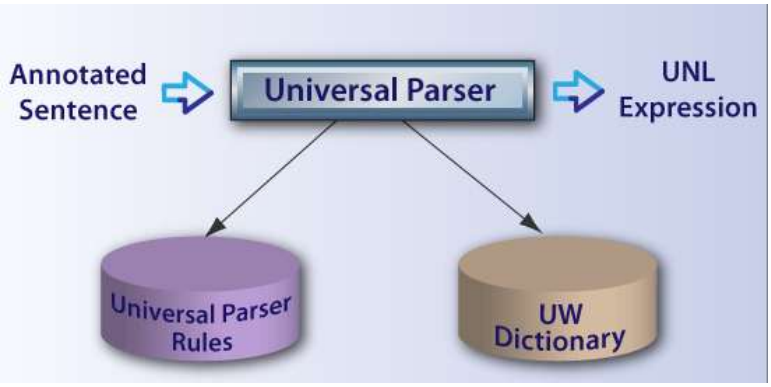

Figure 3.9: Structure of Universal Parser

3.7.8 UNL verifier

The UNL verifier is used to check the syntax, lexical and semantic errors in an input UNL expression. The syntax errors are found on the basis of UNL specifications. The lexical errors are determined on the basis whether all UWs of a UNL expression are defined in the UNL ontology or not. A semantic error is reported if a binary relation of a UNL expression is not defined as per UNL ontology. The flowchart of UNL verifier and use of dictionaries in this are shown in Figure 3.10 (UNDL Foundation, 2010).

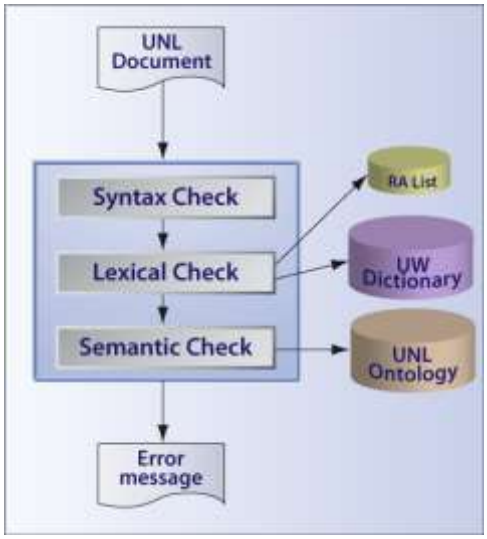

Figure 3.10: Flowchart of UNL verifier

3.7.9 Language Server

The conversion processes between natural languages and UNL expressions are carried on Internet with the help of UNL Language Servers (LSs). Figure 3.11 illustrates working of LS of UNL system (UNDL Foundation, 2010).

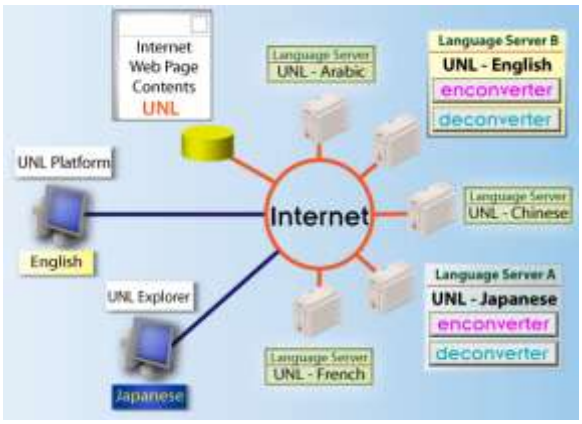

Figure 3.11: Working of LSs over Internet

3.7.10 Word dictionary (Language-UW dictionary)

UNL system makes use of word dictionaries in the form of Language-UW (L-UW) lexicon of respective languages for its processing. It stores information about all the root words of a natural language. An entry of the word dictionary contains three parts, namely, a headword, a UW and a set of morphological, syntactic and semantic attributes. A headword is a word or a morpheme of a natural language. The format for the creation of Language-UW lexicon is given in (3.10).

[HW]{ID} "UW" (ATTR, ATTR,…) <FLG, FRE, PRI>; …(3.10)

The description of these entries is given in Table 3.6.

Table 3.6: Description of Word dictionary entries

HW	Headword of a language, *i.e.*, equivalent Punjabi word to UW
ID	Identification of HW (optional)
UW	Universal Word, can be empty if not necessary
ATTR	Grammatical attributes associated with the HW
FLG	Language flag, *e.g.*, 'P' for Punjabi, 'E' for English, 'H' for Hindi *etc.*
FRE	Frequency of HW
PRI	Priority of HW

The L-UW, i.e., Punjabi-UW dictionary consists of language-dependent and language-independent parts. The Punjabi-UW dictionary stores the HW as root word of a language used to give the meaning to the corresponding UW. For example, for the UW *'boy(icl>male person)'*, the Punjabi word ਮੁੰਡਾ *muṇḍā 'boy'* is the equivalent word. The dictionary stores the root word ਮੁੰਡਾ *muṇḍā* in the lexicon. It represents the noun ending with '◌ਾ' (ਆ) *ā 'A'* that changes its form depending upon the number and case, *i.e.*, ਮੁੰਡਾ *muṇḍā* retains its form for singular-direct and changes to ਮੁੰਡੇ *muṇḍē* for singular-oblique and plural-direct; to ਮੁੰਡਿਆਂ *muṇḍiāṃ* for plural-oblique; to ਮੁੰਡਿਆ *muṇḍiā* for singular-vocative and to ਮੁੰਡਿਓ *muṇḍiō* for plural-vocative case. The Punjabi-UW dictionary will have *'N'* (noun) as its syntactic attribute; *'NA'* (indicating noun word ending with'◌ਾ' (ਆ) *ā 'A'*) as its morphological attribute and *'ANIMATE'* as its semantic attribute as shown in Punjabi-UW dictionary entry in (3.11).

[ਮੁੰਡਾ] {} "boy (icl>male person)" (N, NA, ANIMATE); <P,0,0>;/(*muṇḍā*) …(3.11)

Here, the language-independent parts of this entry are *'boy(icl>male person)'* and *'ANIMATE'*, while the language-dependent parts are *'N'* and *'NA'*.

During the EnConversion process, a word or morpheme of a word dictionary is used as a trigger to obtain an appropriate UW in order to create the UNL expression from an input sentence. During the DeConversion process, word dictionary forms the target sentence of a natural language from a UNL expression. The grammatical attributes of word dictionary are used to define the behavior of word or a morpheme in a sentence. They are used during both EnConversion and DeConversion processes.

3.8 Grammatical attributes of Punjabi-UW lexicon

Punjabi-UW dictionary uses large number of attributes to represent the morphological, syntactic and semantic information about the words. The details of these attributes are given in Table 3.7.

Table 3.7: Some of morphological and semantic attributes used in Punjabi-UW lexicon

Attribute	Description
N	Noun
PROP	Proper noun
MALE	Noun with masculine gender
FEMALE	Noun with feminine gender
1SG	1st person singular
2SG	2nd person singular
3SG	3rd person singular
1PL	1st person plural
2PL	2nd person plural
3PL	3rd person plural
COLCT	Collective noun
NOTCH	Non-changeable, noun that do not change their form with number and case
PRON	Pronoun
ADJ	Adjective
ADV	Adverb
PREP	Preposition
V	Verb
VINT	Intransitive verb
CV	Conjunct verb
link	Verbs that are formed by adding the verb ਕਰ *kar 'do'* to nouns or adjectives
lnk	Verbs that are formed by adding the verb ਹੋ *hō 'be'* to nouns or adjectives
Vowel-ending attributes in Punjabi-UW lexicon	
Na	Noun ending with ਮੁਕਤਾ *muktā* (ਅ) *a*
NA	Noun ending with ਾ (ਆ) *ā*

Ni	Noun ending with ਿ (ਇ) *i*
NI	Noun ending with ੀ (ਈ) *ī*
Nu	Noun ending with (ੁ) *u*
NU	Noun ending with (ੂ) *ū*
Ny	Noun ending with ੇ (ਏ) *ē*
NY	Noun ending with ੈ (ਐ) *ai*
No	Noun ending with ੋ (ੋ) *ō*
NO	Noun ending with ੌ (ਔ) *au*
Va	Verb ending with ਮੁਕਤਾ *muktā* (ਅ) *a*
VA	Verb ending with ਾ (ਆ) *ā*
Vi	Verb ending with ਿ (ਇ) *i*
VI	Verb ending with ੀ (ਈ) *ī*
Vu	Verb ending with (ੁ) *u*
VU	Verb ending with (ੂ) *ū*
Vy	Verb ending with ੇ (ਏ) *ē*
VY	Verb ending with ੈ (ਐ) *ai*
Vo	Verb ending with ੋ (ੋ) *ō*
VO	Verb ending with ੌ (ਔ) *au*
Semantic attributes of noun	
INS	Instrument
ANIMT	Animate
FLORA	Flora
FAUNA	Fauna
BIRD	Bird
IMGYCRT	Imaginary creature
INSCT	Insect
MML	Mammal

INANI	Inanimate
OBJCT	Object
IMGN	Imaginary
PHSCL	Physical
DRNKBL	Drinkable
PLACE	Place
IMGY	Imaginary place
MYTHO	Mythological place
PHSCL	Physical place
ABS	Abstract
ACT	Action
TIME	Time
TITL	Title
GRP	Group
POF	Part of
STE	State
FEEL	Feeling
EMOT	Emotion
Semantic attributes of verbs	
VOA	Verb of action
VOA-CHNG	Change
VOO	Verb of occur
VOS	Verb of state
Semantic attributes of adjectives	
DES	Descriptive
ACT	Action
APPR	Appearance
CLR	Color
DPTH	Depth
DIRCTN	Directional
NUM	Numeral

QUAL	Qualitative
QUAN	Quantitative
RESP	Respective
SHP	Shape
SIZE	Size
SPD	Speed
SMEL	Smell
TSTE	Taste
TEMP	Temperature
TIME	Time
TCH	Touch
WT	Weight
Semantic attributes of adverbs	
INTRO	Interrogative
MAN	Manner
NGTV	Negative

3.9 Important issues in creation of L-UW dictionary

In this section, we present some important issues that have been encountered during the creation of Punjabi-UW dictionary.

- It has been observed that English equivalent of some HWs is not available. If we encounter such a case, where we do not find an equivalent concept in English, then that word is transliterated in *Roman* to use it as headword in L-UW dictionary (Dave *et al.*, 2001). For example, ਵੈਸਾਖੀ, *vaisākhī* 'a Punjab festival' has no equivalent English word or concept. This word is stored in Punjabi-UW dictionary in transliterated form in *Roman* as headword. The disambiguation of such words is performed by adding restrictions or constraints to these headwords. The corresponding entry of this concept in Punjabi-UW dictionary is given in (3.12).

  [ਵੈਸਾਖੀ] {} "vaisakhi(icl>festival)" (N,FEMALE,INANI, NOTCH); <P,00>;

  /(*vaisākhī)* …(3.12)

- If a headword of language does not have any particular sense in the English language, then this sense is also added to UW as its restrictions. For example, one of the senses found in India of the word '*back-bencher'* is 'student who is not serious in his/her studies and passes the time sitting at the back of the class'. This additional sense is included in the UW dictionary as '*back-bencher(icl>student)'*. Thus, if a particular word '*w'* in English has acquired an additional sense in another language; this sense is introduced into the UW dictionary by tagging the appropriate restriction (Dave *et al.*, 2001).
- When an English word has no direct mapping in another language, then English word is transliterated in the target language and stored in the lexicon as language HW (Dave *et al.*, 2001). For example, English word *'printer'* used in the sense of an output device for the computer, has no equivalent Punjabi word, so *'printer' is* transliterated into Punjabi as ਪ੍ਰਿੰਟਰ *priṇṭar* and stored in lexicon as shown in (3.13).

  [ਪ੍ਰਿੰਟਰ] {} "printer(icl>device)" (N,MALE,INANI,Na);<P0,0>;/(*priṇṭar*) …(3.13)

3.10 Creation of Punjabi-UW dictionary

A Punjabi-UW dictionary has been created from the Hindi-UW dictionary (developed by IIT Bombay, India) for the development of UNL based MT system for Punjabi Language. We have created a Punjabi-UW dictionary of 1,15,000 entries for this purpose. The lexicon is stored in the text format. During the development process of Punjabi-UW dictionary, the language-independent component remains the same and language-dependent component like vowel-ending and morphology information are changed according to the Punjabi language. The Hindi headwords are replaced by equivalent Punjabi headwords during this process. A snapshot of Punjabi-UW dictionary is given in (3.14).

[ਅਸਾਧਾਰਣਤਾ]{}"abnormality(icl>phenomenon)" (N,FEMALE,INANI,NA) <P,0,0>; /(*asādhārṇatā*)

[ਦੇ ਬਾਰੇ ਵਿਚ] {}"about(aoj>thing,obj>thing)" (PREP) <P,0,0>; /(*dē bārē vic*)

[ਬਾਬਤ] {}"about(qua<thing)" (PREP) <P,0,0>; /(*bābat*)

[ਲਗਭਗ] {}"about(qua<thing)" (PREP) <P,0,0>; /(*lagbhag*)

[ਵਿਦੇਸ਼] {}"abroad" (N,MALE,INANI,PLACE,PHSCL,Na) <P,0,0>;/(*vidēsh*)

[ਪਹੁੰਚ ਯੋਗ] {}"accessible(aoj>thing,obj>thing)" (ADJ,QUAL) <P,0,0>; /(*pahuñc yōg*)

[ਭਾਗ ਲੈਣਾ] {}"participate(agt>thing,gol>thing)" (V,VA) <P,0,0>;/ (*bhāg laiṇā*)

[ਸਮਾਰੋਹ] {}"party(icl>meeting)" (N,Na,MALE,INANI) <P,0,0>;/(*samārōh*)

[ਲੋਕ] {}"people(icl>person)" (N,3PL,ANIMT,Na) <P,0,0>;/(*lōk*)

[ਸਮਾਂ] {}"period(icl>time)" (N,NOTCH,MALE,INANI) <P,0,0>;/ (*samāṃ*)

[ਅਵੱਲਾ] {}"perverse(aoj>thing)" (ADJ,DES,QUAL,AdjA) <P,0,0>;/(*avllā*)

[ਘਟਨਾ]{}"phenomenon(icl>event)"(N,NOTCH,FEMALE,INANI,NA) <P,0,0>;/ (*ghaṭnā*)

[ਮਸ਼ਹੂਰ] {}"popular(aoj>thing)" (ADJ,QUAL) <P,0,0>; /(*mashhūr*)

[ਮੁਮਕਿਨ] {}"possible(aoj>thing)" (ADJ,DES,QUAL) <P,0,0>;/(*mumkin*)

[ਸ਼ਕਤੀ] {}"power(icl>attribute)" (N,FEMALE,NOTCH,NI) <P,0,0>;/(*shaktī*)

[ਰਾਜ ਕੁਮਾਰ] {}"prince(icl>duke)" (N,MALE,ANIMT,Na) <P,0,0>;/(*rāj kumār*)

[ਉੱਭਰਿਆ]{}"prominent(aoj>thing)" (ADJ,AdjA) <P,0,0>;/(*ubbhriā*)

[ਜਨਤਾ] {}"public(icl>person)" (N,NOTCH,FEMALE,ANIMT,NA) <P,0,0>;/(*jantā*)

[ਪ੍ਰਕਾਸ਼ਨ] {}"publishing(icl>action)"(N,MALE,INANI,NOTCH,Na)<P,0,0>; /(*prakāshan*)

[ਪ੍ਰਾਪਤ] {}"receive(agt>thing,obj>thing)" (V,Va) <P,0,0>;/(*prāpat*)

[ਨਾਤਾ] {}"relationship(icl>relation)" (N,MALE,INANI,NA) <P,0,0>;/(*nātā*)

[ਮੁਕਤੀ] {}"salvation(icl>action)" (N,NOTCH,FEMALE,INANI,ABS,NI)<P,0,0>; /(*muktī*)

[ਸਮਾਨ] {}"same(mod<thing)" (ADJ,QUAL) <P,0,0>; /(*samān*)

[ਉਵੇਂ ਹੀ] {}"similarly" (ADV,MAN) <P,0,0>;/(*uvēṃ hī*)

[ਸਰੋਤ] {}"source(icl>origin)" (N,MALE,INANI,Na) <P,0,0>;/(*sarōt*)

[ਬੋਲ] {}"speak(agt>thing,obj>language)" (V,Va) <P,0,0>;/(*bōl*)

[ਚੱਮਚ] {}"spoon" (N,INS,INANI,Na) <P,0,0>;/(*cammac*)

[ਕੰਪਿਊਟਰ]{}"computer(icl>machine)"(N,MALE,INANI,MACH,Na)<P,0,0>; /(*kampiūṭar*)

[ਪੰਜਾਬੀ]{}"punjabi(icl>language)"(N,NOTCH,FEMALE,INANI)<P,0,0>
;/(*pañjābī*) …(3.14)

Chapter Summary

Universal Networking Language (UNL) is an interligual representation of knowledge. It produces a representation for semantically equivalent sentences of all languages. The EnConverter and DeConverter are the core softwares in a UNL system. EnConverter converts source language sentences into UNL expressions and DeConverter converts UNL expressions to target language sentences. UNL represents the information in three different types of semantic units, namely, Universal Words (UWs), Relations and Attributes. UNL represents information sentence by sentence. Each sentence can be converted into a UNL graph having concepts as nodes and relations as directed arcs. The UWs are divided into four types: Basic UWs; Restricted UWs; Extra UWs and Temporary UWs. Compound UWs denote compound concepts that are to be interpreted as a whole so that one can talk about their parts all at the same time. A compound UW is expressed by a scope in UNL expressions. A UNL sentence can be represented either in table format or in list format. UNL knowledge-base is used to define every possible relation between concepts. A UNL system consists of three major components, namely, language resources, software for processing language resources, and supporting tools for maintaining and operating language processing software. UNL system makes use of word dictionaries in the form of Language-UW lexicon of respective languages for its processing. An entry of the word dictionary contains three parts, namely, a headword, a UW and a set of morphological, syntactic and semantic attributes. A Punjabi-UW dictionary has been created from the Hindi-UW dictionary (developed by IIT Bombay, India) for the development of UNL based MT system for Punjabi Language during the progress of this work.

Chapter 4

Punjabi-UNL EnConverter

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.

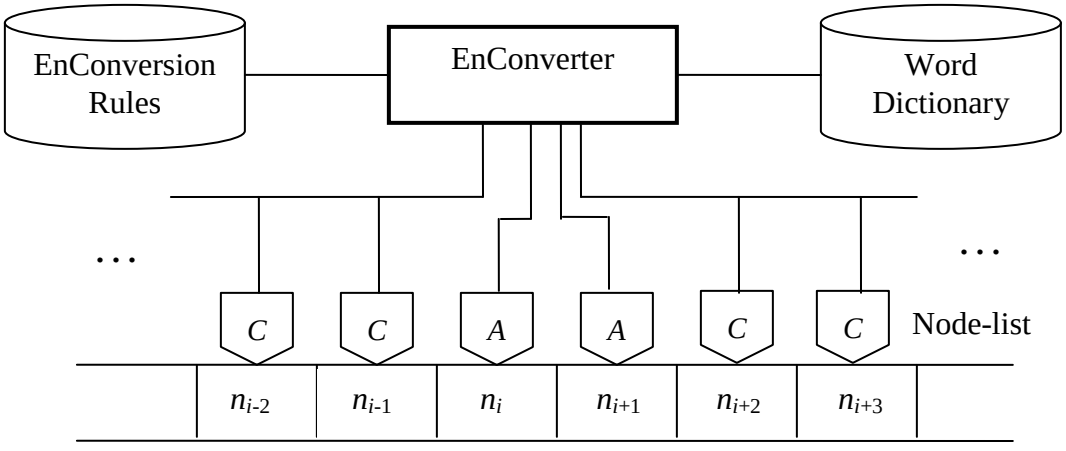

Figure 4.1: Analysis and condition windows of Punjabi EnConverter

In Figure 4.1, *'A'* indicates an analysis window, *'C'* indicates a condition window, and 'n_i' 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} …(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, *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,INS:null:ins}{V:+INSRES:null} …(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 ਬੱਚਿਆਂ ਨੇ ਕਿਤਾਬ ਪੜ੍ਹੀ। …(4.3)

Transliterated example sentence: *chōṭē bacciāṃ nē kitāb paṛhī.*

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:

<Sentence id="1">

1 ਛੋਟੇ unk

2 ਬੱਚਿਆਂ unk

3 ਨੇ unk

4 ਕਿਤਾਬ unk

5 ਪੜ੍ਹੀ unk

</Sentence> …(4.4)

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:

<Sentence id="1">

1 ਛੋਟੇ unk <fs af='ਛੋਟਾ,adj,m,sg,,o,,'>|

<fs af='ਛੋਟਾ,adj,m,pl,,o,,'>

2 ਬੱਚਿਆਂ 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 ਕਿਤਾਬ unk <fs af='ਕਿਤਾਬ,n,f,sg,3,d,,'>

5 ਪੜ੍ਹੀ unk <fs af='ਪੜ੍ਹ,v,f,sg,any,,ਇਆ,ਇਆ'>

</Sentence> …(4.5)

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), ਛੋਟੇ *chōṭē* has a root word ਛੋਟਾ *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 ਛੋਟੇ JJ <fs af='ਛੋਟਾ,adj,m,sg,,o,,'>|

<fs af='ਛੋਟਾ,adj,m,pl,,o,,'>

2 ਬੱਚਿਆਂ NN <fs af='ਬੱਚਾ,n,m,pl,3,o,,'>

3 ਨੇ PSP <fs af='ਨੇ,psp,,,,d,,'>|<fs af='ਆਂ,v,any,pl,2,,,'>

4 ਕਿਤਾਬ NN <fs af='ਕਿਤਾਬ,n,f,sg,3,d,,'>

5 ਪੜ੍ਹੀ VM <fs af='ਪੜ੍ਹ,v,f,sg,any,,ਇਆ,ਇਆ'>

</Sentence> …(4.6)

As shown in (4.6), ਛੋਟੇ *chōṭē 'little'* is an adjective with symbol *'JJ'*, ਬੱਚਿਆਂ *bacciāṃ 'children'* and ਕਿਤਾਬ *kitāb 'book'* are nouns with symbol *'NN'*, ਨੇ *nē* is postposition with symbol *'PSP'* and ਪੜ੍ਹੀ *paṛhī '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 ਬੱਚਿਆਂ NN <fs af='ਬੱਚਾ,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 ਕਿਤਾਬ NN <fs af='ਕਿਤਾਬ,n,f,sg,3,d,,'>

))

3 ((VGF

3.1 ਪੜ੍ਹੀ VM <fs af='ਪੜ੍ਹ,v,f,sg,any,,ਇਆ,ਇਆ'>

))

</Sentence> …(4.7)

As shown in (4.7), the example sentence has three chunks, ‘ਛੋਟੇ ਬੱਚਿਆਂ ਨੇ’ *‘chōṭ̱ē bacciāṃ nē’ ‘liitle children’* and ਕਿਤਾਬ *kitāb ‘book’* as noun phrase *‘NP’* chunks and ਪੜ੍ਹੀ *paṛ̱hī ‘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).

<Sentence id="1">

1 ((NP

1.1 ਛੋਟੇ JJ <fs af='ਛੋਟਾ,adj,m,sg,,o,,'>

1.2 ਬੱਚਿਆਂ NN <fs af='ਬੱਚਾ,n,m,pl,3,o,,'>

1.3 ਨੇ PSP <fs af='ਨੇ,psp,,,,d,,'>

))

2 ((NP

2.1 ਕਿਤਾਬ NN <fs af='ਕਿਤਾਬ,n,f,sg,3,d,,'>

))

3 ((VGF

3.1 ਪੜ੍ਹੀ VM <fs af='ਪੜ੍ਹ,v,f,sg,any,,ਇਆ,ਇਆ'>

))

</Sentence> …(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).

<Sentence id="1">

1 ((NP <fs af='ਬੱਚਾ,n,m,pl,3,o,,' head="ਬੱਚਿਆਂ">

1.1 ਛੋਟੇ JJ <fs af='ਛੋਟਾ,adj,m,sg,,o,,'>

1.2 ਬੱਚਿਆਂ NN <fs af='ਬੱਚਾ,n,m,pl,3,o,,' name="ਬੱਚਿਆਂ">

1.3 ਨੇ PSP <fs af='ਨੇ,psp,,,,d,,'>

))

2 ((NP <fs af='ਕਿਤਾਬ,n,f,sg,3,d,,' head="ਕਿਤਾਬ">

2.1 ਕਿਤਾਬ NN <fs af='ਕਿਤਾਬ,n,f,sg,3,d,,' name="ਕਿਤਾਬ">

))

3 ((VGF <fs af='ਪੜ੍ਹ,v,f,sg,any,,ਇਆ,ਇਆ' head="ਪੜ੍ਹੀ">

3.1 ਪੜ੍ਹੀ VM <fs af='ਪੜ੍ਹ,v,f,sg,any,,ਇਆ,ਇਆ' name="ਪੜ੍ਹੀ">

))

</Sentence> ...(4.9)

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).

<Sentence id="1">

1 ((NP <fs af='ਬੱਚਾ,n,m,pl,3,o,0_ਨੇ,' head="ਬੱਚਿਆਂ vpos="vib2_3">

1.1 ਛੋਟੇ JJ <fs af='ਛੋਟਾ,adj,m,sg,,o,,'>

1.2 ਬੱਚਿਆਂ NN <fs af='ਬੱਚਾ,n,m,pl,3,o,,' name="ਬੱਚਿਆਂ">

))

2 ((NP <fs af='ਕਿਤਾਬ,n,f,sg,3,d,,' head="ਕਿਤਾਬ">

2.1 ਕਿਤਾਬ NN <fs af='ਕਿਤਾਬ,n,f,sg,3,d,,' name="ਕਿਤਾਬ">

))

3 ((VGF <fs af='ਪੜ੍ਹ,v,f,sg,any,,ਇਆ,ਇਆ' head="ਪੜ੍ਹੀ">

3.1 ਪੜ੍ਹੀ VM <fs af='ਪੜ੍ਹ,v,f,sg,any,,ਇਆ,ਇਆ' name="ਪੜ੍ਹੀ">

))

</Sentence> …(4.10)

As shown in (4.10), in chunk-1, function word ਨੇ *nē* is grouped with the *'head'* word ਬੱਚਿਆਂ *bacciāṃ 'children'* with *'vibhakti' 'vib2_3'* and in chunk-3, verb modifier ਇਆ *iā* is grouped with *'head'* word ਪੜ੍ਹੀ *paṛhī '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.*, 'ਸਯੁਕਤ ਰਾਸ਼ਟਰ ਅਮਰੀਕਾ' *'sayukat rāshṭar amrīkā' 'United States of America'*) or on the basis of groups of words that modify the root verb (*e.g.*, 'ਰਹੀ ਹੈ', *'rahī hai'*).

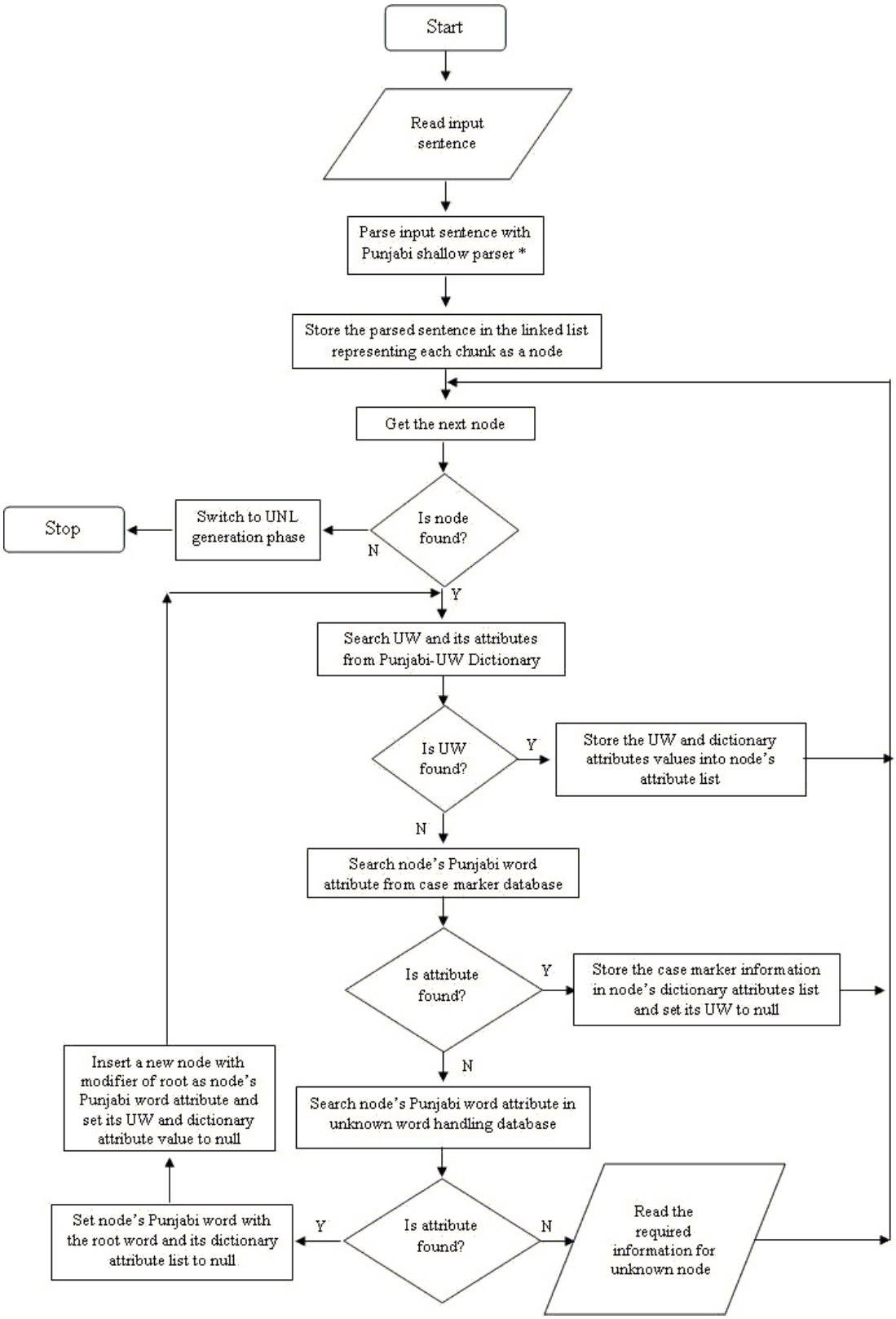

Figure 4.2: Flowchart for working of Punjabi EnConverter

* 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.

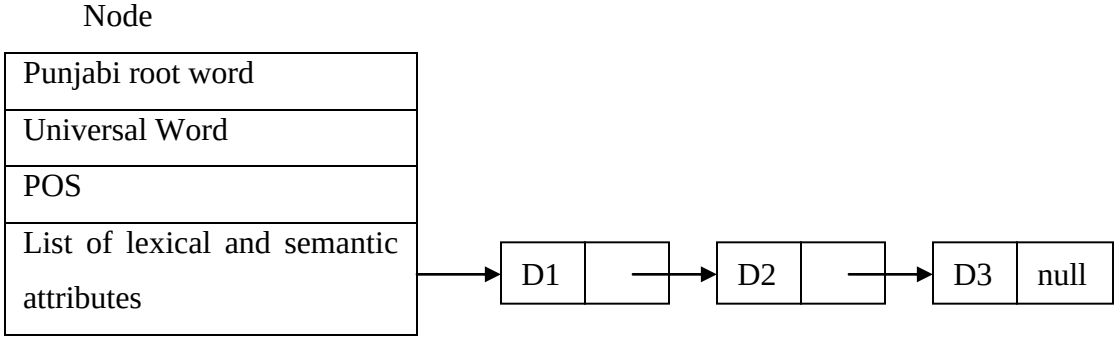

Figure 4.3: Structure of a node

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 *chōṭā 'little'*, ਬੱਚਾ *baccā 'child'*, ਨੇ *nē*, ਕਿਤਾਬ *kitāb 'book'*, ਪੜ੍ਹ *paṛh 'read'*, ਇਆ *iā*. For the first node, words ਛੋਟਾ *chōṭā*, 'ਛੋਟਾ ਬੱਚਾ' '*chōṭā baccā'*, 'ਛੋਟਾ ਬੱਚਾ ਨੇ' '*chōṭā baccā nē'*, 'ਛੋਟਾ ਬੱਚਾ ਨੇ ਕਿਤਾਬ' '*chōṭā baccā nē kitāb'* are searched in Punjabi-UW dictionary and verb-modifier database. In this case, a node will be created for the token, ਛੋਟਾ *chōṭā 'little'*, in the linked list, because it has an entry in Punjabi-UW dictionary. Similarly, the nodes are created for ਬੱਚਾ *baccā 'child'*, ਨੇ *nē*, ਕਿਤਾਬ *kitāb 'book'*, ਪੜ੍ਹ *paṛh 'read'* and ਇਆ *iā*. Thus, example sentence given in (4.3), results into creation of linked list of six nodes, namely, ਛੋਟਾ *chōṭā 'little'*, ਬੱਚਾ *baccā 'child'*, ਨੇ *nē*, ਕਿਤਾਬ *kitāb 'book'*, ਪੜ੍ਹ *paṛh '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).

$Node_1$ Attributes: Punjabi root word: ਛੋਟਾ *chōṭā*; UW: little; POS: ADJ; Lexical and semantic attribute list: m, sg, o, ADJ.

$Node_2$ Attributes: Punjabi root word: ਬੱਚਾ *baccā*; UW: child; POS: NN; Lexical and semantic attribute list: n, m, pl, 3, o, N, ANIMT.

$Node_3$ Attributes: Punjabi root word: ਨੇ *nē*; UW: null; POS: PSP; Lexical and semantic attribute list: psp, d.

$Node_4$ Attributes: Punjabi root word: ਕਿਤਾਬ *kitāb*; UW: book; POS: NN; Lexical and semantic attribute list: n, f, sg, 3, d, N, INANI.

$Node_5$ Attributes: Punjabi root word: ਪੜ੍ਹ *paṛh*; UW: study(icl>do); POS: VM; Lexical and semantic attribute list: v, f, sg, any, V.

$Node_6$ Attributes: Punjabi root word: ਇਆ *iā*; UW: null; POS: null; Lexical and semantic attribute list: null. …(4.11)

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.,*ਜਾਵਾਂਗਾ *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, ਜਾਵਾਂਗਾ *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 ਜਾਵਾਂਗਾ *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<Sentence id="1">

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">

))

</Sentence> …(4.13)

ii) Linked list creation phase

In this phase, tokens ਮੈਂ *maiṃ ‘i’*, ਗੈਰੇਜ *gairēj ‘garage’*, ਵਿਚ *vic ‘in’*, ਕਾਰ *kār ‘car’*, ਧੋ *dhō ‘wash’*, ਦਾ *dā* and ਹਾਂ *hāṃ* 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 *maiṃ*; UW: null; POS: PRP; Lexical and semantic attribute list: pn, any, sg, 1, d.

$Node_2$ Attributes: Punjabi root word: ਗੈਰੇਜ *gairēj*; UW: null; POS: NN; Lexical and semantic attribute list: unk.

$Node_3$ 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: null; POS: NN; Lexical and semantic attribute list: n, f, sg, 3, d.

$Node_5$ Attributes: Punjabi root word: ਧੋ *dhō*; UW: null; POS: VM; Lexical and semantic attribute list: v, m, sg, any.

$Node_6$ Attributes: Punjabi root word: ਦਾ ਹਾਂ *dā hāṃ*; UW: null; POS: null; Lexical and semantic attribute list: n, f, sg, 3, d. …(4.14)

It has been noted that $Node_2$ corresponding to Punjabi word ਗੈਰੇਜ *gairēj ‘garage’* is classified into an unknown node, since the word is not resolved by Punjabi shallow parser. $Node_6$ contains two Punjabi words ਦਾ *dā* and ਹਾਂ *hāṃ*, 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 *maiṃ*; UW: I(icl<person)); POS: PRP; Lexical and semantic attribute list: pn, any, sg, 1, d, PERPRON, ANIMT.

$Node_2$ Attributes: Punjabi root word: ਗੈਰੇਜ *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 ਹਾਂ *dā hāṃ*; UW: null; POS: null; Lexical and semantic attribute list t: n, f, sg, 3, d. …(4.15)

The unknown word ਗੈਰੇਜ *gairēj 'garage'* in $Node_2$ becomes known in this phase as ਗੈਰੇਜ *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).

<<[ਮੈਂ] [ਗੈਰੇਜ] ਵਿਚ ਕਾਰ ਧੋ ਦਾ ਹਾਂ>> …(4.16)

<<[maiṃ] [gairēj] vic kār dhō dā hāṃ>>

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).

<<ਮੈਂ [ਗੈਰੇਜ] [ਵਿਚ] ਕਾਰ ਧੋ ਦਾ ਹਾਂ>> …(4.17)

<<maiṃ [gairēj] [vic] kār dhō dā hāṃ>>

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

+{N,PLC:+WICH:null}{[ਵਿਚ]: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 ਵਿਚ *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).

<<[ਮੈਂ] [ਗੈਰੇਜ_ਵਿਚ] ਕਾਰ ਧੋ ਦਾ ਹਾਂ>> …(4.19)

<<[maiṃ] [gairēj_ vic] kār dhō dā hāṃ>>

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).

<<ਮੈਂ [ਗੈਰੇਜ_ਵਿਚ] [ਕਾਰ] ਧੋ ਦਾ ਹਾਂ>> …(4.20)

<<maiṃ [gairē_ vic] [kār] dhō dā hāṃ>>

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).

<<ਮੈਂ ਗੈਰੇਜ_ਵਿਚ [ਕਾਰ] [ਧੋ] ਦਾ ਹਾਂ>> …(4.21)

<<maiṃ gairēj_vic [kār] [dhō] dā hāṃ>>

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

>{N,INANI,^WICH,^PLC,^SRCRES,^TOON:null:obj}{V:+OBJRES:null} …(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).

obj(wash(icl>do), car(icl>thing)) …(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).

<<[ਮੈਂ] [ਗੈਰੇਜ_ਵਿਚ] ਧੋ ਦਾ ਹਾਂ>> …(4.24)

<<[maiṃ] [gairēj_vic] dhō dā hāṃ>>

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).

<<ਮੈਂ [ਗੈਰੇਜ_ਵਿਚ] [ਧੋ] ਦਾ ਹਾਂ>> …(4.25)

<<maiṃ [gairēj_vic] [dhō] dā hāṃ>>

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

>{N,PLC,WICH:null:plc}{V:null:null} …(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.

plc(wash(icl>do),garage(icl>thing)) …(4.27)

The node-list now becomes as shown in (4.28).

<<[ਮੈਂ] [ਧੋ] ਦਾ ਹਾਂ>> …(4.28)

<<[maiṃ] [dhō] dā hāṃ>>

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

>{PERPRON,ANIMT:null:agt}{V:+AGTRES:null} …(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.

agt(wash(icl>do), I(icl<person)) …(4.30)

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

<<[ਧੋ] [ਦਾ ਹਾਂ]>> …(4.31)

<<[dhō] [dā hāṃ]>>

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

+{V:+.@present.@custom.@sg.@male:null}{[ਦਾ ਹਾਂ]:null:null} …(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 'ਦਾ ਹਾਂ' *'dā hāṃ'* in '[' and ']', it means that the rule will fire if the right analysis window has Punjabi root word attribute 'ਦਾ ਹਾਂ' *'dā hāṃ'*. 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).

<<[ਧੋ_ਦਾ ਹਾਂ]>> …(4.33)

<<[dhō_dā hāṃ]>>

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).

wash(icl>do).@present.@custom.@sg.@male.@entry …(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 ਗੈਰੇਜ ਵਿਚ ਕਾਰ ਧੋਂਦਾ ਹਾਂ ।

Transliterated sentence: *maiṃ gairēj vic kār dhōndā hāṃ.*

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.

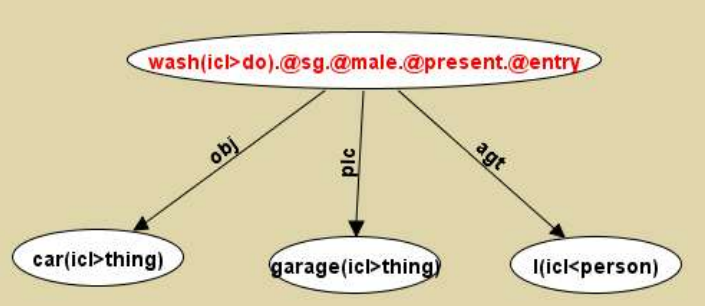

Figure 4.4: UNL graph for the UNL expression in (4.35)

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 ਕਿ *ki '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 ਕਿਹਾ ਕਿ ਸੁੱਖਵਿੰਦਰ ਚੰਗਾ 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,* 'ਕਹਿ ਇਆ' *'kahi iā' 'said',* ਕਿ *ki 'that',* ਸੁੱਖਵਿੰਦਰ *sukkhvindar 'Sukhwinder'*, ਚੰਗਾ *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] ਕਹਿ ਇਆ ਕਿ ਸੁੱਖਵਿੰਦਰ ਚੰਗਾ 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] [ਕਹਿ ਇਆ] ਕਿ ਸੁੱਖਵਿੰਦਰ ਚੰਗਾ 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:null:agt}

{V,^AGTRES,^VOCURR,^COO:+AGTRES: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).

<<[ਕਹਿ ਇਆ] [ਕਿ] ਸੁੱਖਵਿੰਦਰ ਚੰਗਾ ਹੈ>> …(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).

<<ਕਹਿ ਇਆ [ਕਿ] [ਸੁੱਖਵਿੰਦਰ] ਚੰਗਾ ਹੈ>> …(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.

-{[ਕਿ]: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).

<<[ਕਹਿ ਇਆ] [ਕਿ_ਸੁੱਖਵਿੰਦਰ] ਚੰਗਾ ਹੈ>> …(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).

<<ਕਹਿ ਇਆ [ਕਿ_ਸੁੱਖਵਿੰਦਰ] [ਚੰਗਾ] ਹੈ>> …(4.46)

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

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

>{N,^TIME,^TOON,^WICH,NCL:null:aoj}{ADJ,ST:+AOJRES,+NCL:null} …(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.

aoj(good(icl>state), Sukhwinder(icl>person)) …(4.48)

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

<<[ਕਹਿ ਇਆ] [ਚੰਗਾ] ਹੈ>> …(4.49)

<<[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:+.@entry:null}{ADJ,NCL:+COMPLEX:obj} …(4.50)

obj(say.@entry, good) …(4.51)

The rule given in (4.50) resolves a UNL relation *'obj'* between main verb and sub-ordinate 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).

obj(say.@entry, :01) …(4.52)

aoj:01(good, Sukhwinder(icl>person)) …(4.53)

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

<<[ਚੰਗਾ] [ਹੈ]>> …(4.54)

<<[caṅgā] [hai]>>

Here, left composition rule given in (4.55) is fired between left and right analysis windows.

+{ADJ:+VAUX,+.@present.@sg:null}{[ਹੈ]:null:null} …(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).

<<[ਚੰਗਾ_ਹੈ]>> …(4.56)

<<[caṅgā_hai]>>

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} …(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.

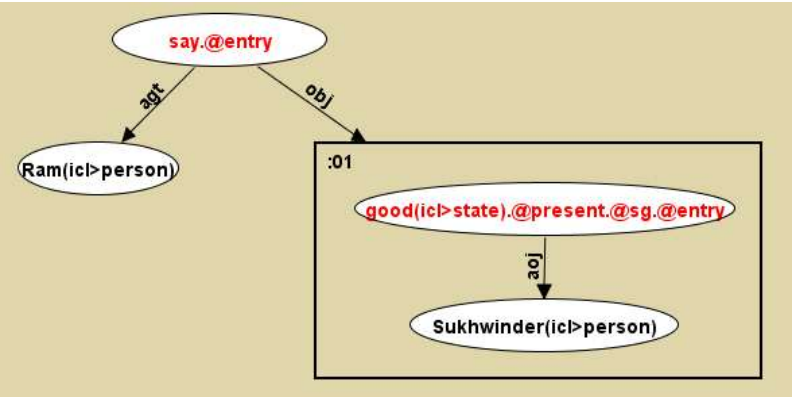

Figure 4.5: UNL graph for UNL expression given in (4.57)

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 4.1: Types of adverb clause

Type of clause	Attributes used	Punjabi words used to represent adverb clause
Time Adverb Clause	TCLB, TCL	ਜਦੋਂ *jadōṃ 'when'* and ਓਦੋਂ *ōdōṃ 'then'*
Conditional Adverb Clause	CCLB, CCL	ਜੇਕਰ/ਜੇ *jēkar/jē 'if'* and ਤਾਂ/ਤੇ *tāṃ/tē 'then'*
Manner Adverb Clause	MCLB, MCL	ਜਿੱਦਾਂ *jiddāṃ* and ਓਦਾਂ *ōdāṃ*
Place Adverb Clause	PCLB, PCL	ਜਿੱਥੇ *jitthē* and ਉੱਥੇ *utthē*

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 ਜਦੋਂ *jadōṃ 'when'* and ਓਦੋਂ *ōdōṃ 'then'* act as the clause delimiter for adverb clause for time in Punjabi. The word, ਜਦੋਂ *jadōṃ 'when'*, indicates the beginning of time condition and is represented by *'TCLB'* attribute, while word, ਓਦੋਂ *ōdōṃ '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: ਜਦੋਂ ਮੈਂ ਕਾਲਜ ਤੋਂ ਆਵਾਂਗਾ ਓਦੋਂ ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾਵੇਂਗਾ । …(4.58)

Transliterated Punjabi sentence: *jadōṃ maiṃ kālaj tōṃ āvāṅgā ōdōṃ 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)

Iteration:1	Node-net	[ਜਦੋਂ] [ਮੈਂ] ਕਾਲਜ ਤੋਂ ਆ ਏਗਾ ਓਦੋਂ ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ *[jadōṃ] [maiṃ] kālaj tōṃ ā ēgā ōdōṃ tūṃ bāzār nūṃ jā ēgā*
	Rule fired	-{TCLB,^V,^N,^PRON,^ADJ,^ADV:null:null} {PERPRON:+TCLB:null}
	Action taken	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.
Iteration:2	Node-net	[ਜਦੋਂ_ਮੈਂ][ਕਾਲਜ] ਤੋਂ ਆ ਏਗਾ ਓਦੋਂ ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ

		[jadōṃ_maiṃ] [kālaj] tōṃ ā ēgā ōdōṃ tūṃ bāzār nūṃ jā ēgā
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:3	Node-net	ਜਦੋਂ_ਮੈਂ [ਕਾਲਜ][ਤੋਂ] ਆ ਏਗਾ ਓਦੋਂ ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ *jadōṃ_maiṃ [kālaj] [tōṃ] ā ēgā ōdōṃ tūṃ bāzār nūṃ jā ēgā*
	Rule fired	+{N:+TOON:null}{[ਤੋਂ]: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 *'TOON'* attribute in lexical semantic attribute list of node at left analysis window.
Iteration:4	Node-net	[ਜਦੋਂ_ਮੈਂ] [ਕਾਲਜ_ਤੋਂ] ਆ ਏਗਾ ਓਦੋਂ ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ *[jadōṃ_maiṃ] [kālaj_ tōṃ] ā ēgā ōdōṃ tūṃ bāzār nūṃ jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:5	Node-net	ਜਦੋਂ_ਮੈਂ [ਕਾਲਜ_ਤੋਂ] [ਆ] ਏਗਾ ਓਦੋਂ ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ *jadōṃ_maiṃ [kālaj_ tōṃ] [ā] ēgā ōdōṃ tūṃ bāzār nūṃ jā ēgā*
	Rule fired	>{N,TOON,PLC,^FMTRES,^TIME,^DUR:null:plf} {V:+PLFRES: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, *'plf'* relation is resolved between

		two analysis windows as shown below. plf(come(icl>do), college(icl>school))
Iteration:6	Node-net	[ਜਦੋਂ_ਮੈਂ] [ਆ] ਏਗਾ ਓਦੋਂ ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ *[jadōṃ_maiṃ] [ā] ēgā ōdōṃ tūṃ bāzār nūṃ jā ēgā*
	Rule fired	>{PERPRON,ANIMT,TCLB,^CAG,^NOO:null:agt} {V,^VAUX,^AGTRES,^VOCURR,^COO:+AGTRES, +TCLB: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, *'agt'* relation is resolved between two analysis windows as shown below. agt(come(icl>do), I(icl>person))
Iteration:7	Node-net	[ਆ] [ਏਗਾ] ਓਦੋਂ ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ *[ā][ēgā] ōdōṃ tūṃ bāzār nūṃ jā ēgā*
	Rule fired	+{V,m,sg:+.@future.@sg.@male:null}{[ਏਗਾ]: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 *'come(icl>do)'* is modified as *'come(icl>do).@future.@sg.@male'*.
Iteration:8	Node-net	[ਆ_ਏਗਾ] [ਓਦੋਂ] ਤੂੰ ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ *[ā_ ēgā] [ōdōṃ] tūṃ bāzār nūṃ jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:9	Node-net	ਆ_ਏਗਾ [ਓਦੋਂ] [ਤੂੰ] ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ

		ā_ ēgā [ōdōṃ] [tūṃ] bāzār nūṃ jā ēgā
	Rule fired	-{TCL,^V,^N,^PRON,^ADJ,^ADV:null:null} {PRON:+TCL:null}
	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	[ਆ_ਏਗਾ] [ਓਦੋਂ_ਤੂੰ] ਬਾਜ਼ਾਰ ਨੂੰ ਜਾ ਏਗਾ *[ā_ ēgā] [ōdōṃ_tūṃ] bāzār nūṃ jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:11	Node-net	ਆ_ਏਗਾ [ਓਦੋਂ_ਤੂੰ] [ਬਾਜ਼ਾਰ] ਨੂੰ ਜਾ ਏਗਾ *ā_ ēgā [ōdōṃ_tūṃ] [bāzār] nūṃ jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:12	Node-net	ਆ_ਏਗਾ ਓਦੋਂ_ਤੂੰ [ਬਾਜ਼ਾਰ] [ਨੂੰ] ਜਾ ਏਗਾ *ā_ ēgā ōdōṃ_tūṃ [bāzār] [nūṃ] jā ēgā*
	Rule fired	+{N:+NOO:null}{[ਨੂੰ]: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.
Iteration:13	Node-net	[ਆ_ਏਗਾ] [ਓਦੋਂ_ਤੂੰ] ਬਾਜ਼ਾਰ_ਨੂੰ ਜਾ ਏਗਾ *[ā_ ēgā] [ōdōṃ_tūṃ] bāzār nūṃ jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:14	Node-net	ਆ_ਏਗਾ [ਓਦੋਂ_ਤੂੰ] [ਬਾਜ਼ਾਰ_ਨੂੰ] ਜਾ ਏਗਾ *ā_ ēgā [ōdōṃ_tūṃ] [bāzār_ nūṃ] jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:15	Node-net	ਆ_ਏਗਾ ਓਦੋਂ_ਤੂੰ [ਬਾਜ਼ਾਰ_ਨੂੰ] [ਜਾ] ਏਗਾ *ā_ ēgā ōdōṃ_tūṃ [bāzār_ nūṃ] [jā] ēgā*
	Rule fired	>{N,PLC:null:plt}{V,jA:+PLTRES: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, *'plt'* relation is resolved between two analysis windows as shown below. plt(go(icl>do), market)
Iteration:16	Node-net	[ਆ_ਏਗਾ] [ਓਦੋਂ_ਤੂੰ] ਜਾ ਏਗਾ *[ā_ ēgā] [ōdōṃ_tūṃ] jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:17	Node-net	ਆ_ਏਗਾ [ਓਦੋਂ_ਤੂੰ] [ਜਾ] ਏਗਾ

		ā_ ēgā [ōdōṃ_tūṃ] [jā] ēgā
	Rule fired	>{PRON,ANIMT,TCL,^CAG,^NOO,^PSP,sg,^CCLB, ^CCL:null:agt} {V,^VAUX,^AGTRES,^VOCURR,^COO:+AGTRES, +TCL: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, *'agt'* relation is resolved between two analysis windows as shown below. agt(go(icl>do), you(icl>person))
Iteration:18	Node-net	[ਆ_ਏਗਾ] [ਜਾ] ਏਗਾ *[ā_ ēgā] [jā] ēgā*
	Rule fired	>{V,TCLB:+.@entry:null}{V,TCL:null:tim}
	Action taken	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>thing), :01) plt:01(go(icl>do), market) agt:01(go(icl>do), you(icl>person))

Iteration:19	Node-net	[ਜਾ] [ਏਗਾ] *[jā] [ēgā]*
	Rule fired	+{V,m,sg:+.@future.@sg.@male:null}{[ਏਗਾ]: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 *'go(icl>do)'* is modified as *'go(icl>do).@future.@sg.@male'*.
Iteration:20	Node-net	[ਜਾ_ਏਗਾ] *[jā_ēgā]*
	Rule fired	No rule fired.
	Action taken	Since, it is the last node so *'.@entry'* will be added to UW to *'go(icl>do).@future.@sg.@male'* and the UNL generation process completes at this iteration.

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.

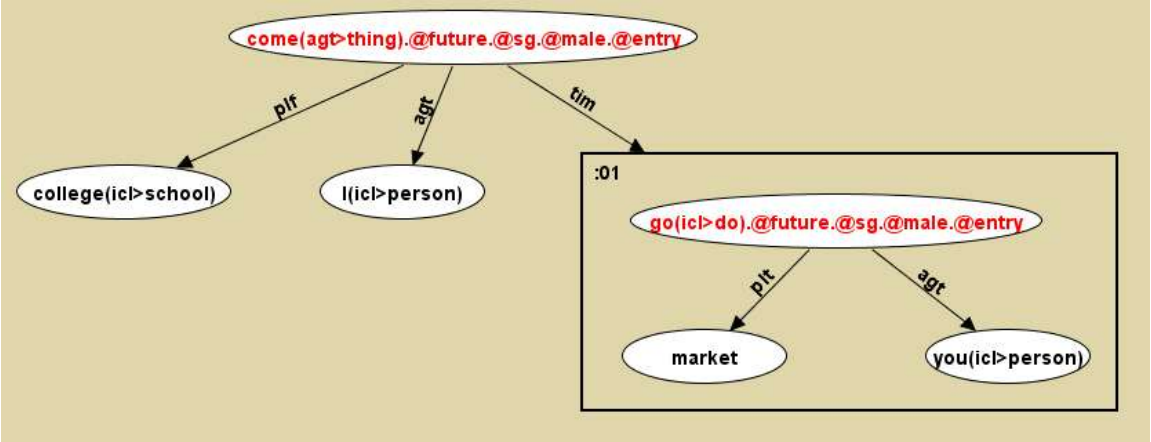

Figure 4.6: UNL Graph for UNL expression given in (4.59)

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 ਸੁਝਾਅ ਨਹੀਂ ਮੰਨਿਆ ਤਾਂ ਤੂੰ ਪਛਤਾਵੇਂਗਾ । …(4.60)

Transliterated Punjabi sentence: *jēkar tūṃ mērā sujhāa nahīṃ manniā tāṃ 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)

Iteration:1	Node-net	[ਜੇਕਰ] [ਤੂੰ] ਮੈਂ ਸੁਝਾਅ ਨਹੀਂ ਮੰਨ ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *[jēkar] [tūṃ] maiṃ sujhāa nahīṃ mann iā tāṃ tūṃ pachtā ēgā*
	Rule fired	-{CCLB,^V,^N,^PRON,^ADJ,^ADV:null:null} {PRON:+CCLB:null}
	Action taken	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.
Iteration:2	Node-net	[ਜੇਕਰ_ਤੂੰ] [ਮੈਂ] ਸੁਝਾਅ ਨਹੀਂ ਮੰਨ ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *[jēkar_tūṃ] [maiṃ] sujhāa nahīṃ mann iā tāṃ tūṃ pachtā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:3	Node-net	ਜੇਕਰ_ਤੂੰ [ਮੈਂ] [ਸੁਝਾਅ] ਨਹੀਂ ਮੰਨ ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *jēkar_tūṃ [maiṃ] [sujhāa] nahīṃ mann iā tāṃ tūṃ pachtā ēgā*
	Rule fired	>{PRON,POS:null:mod}

		{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	[ਜੇਕਰ_ਤੂੰ] [ਸੁਝਾਅ] ਨਹੀਂ ਮੰਨ ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *[jēkar_tūṃ] [sujhāa] nahīṃ mann iā tāṃ tūṃ pachtā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:5	Node-net	ਜੇਕਰ_ਤੂੰ [ਸੁਝਾਅ] [ਨਹੀਂ] ਮੰਨ ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *jēkar_tūṃ [sujhāa] [nahīṃ] mann iā tāṃ tūṃ pachtā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:6	Node-net	ਜੇਕਰ_ਤੂੰ ਸੁਝਾਅ [ਨਹੀਂ] [ਮੰਨ] ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *jēkar_tūṃ sujhāa [nahīṃ] [mann] iā tāṃ tūṃ pachtā ēgā*
	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	[ਜੇਕਰ_ਤੂੰ] [ਸੁਝਾਅ] ਮੰਨ ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *[jēkar_tūṃ] [sujhāa] mann iā tāṃ tūṃ pachtā ēgā*
	Rule fired	No rule fired.

	Action taken	The analysis windows are shifted to right.
Iteration:8	Node-net	ਜੇਕਰ_ਤੂੰ [ਸੁਝਾਅ] [ਮੰਨ] ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *jēkar_tūṃ [sujhāa] [mann] iā tāṃ tūṃ pachtā ēgā*
	Rule fired	>{N,INANI,^MACH,^WICH,^PLC,^RSN,^SRCRES, ^TOON,^TAK,^NOO,^ben/pur/to,^AOJ,^plc/scn,^ACT :null:obj}{V,^WICH:+OBJRES: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, *'obj'* relation is resolved between two analysis windows as shown below. obj(agree(icl>do).@not, suggestion)
Iteration:9	Node-net	[ਜੇਕਰ_ਤੂੰ] [ਮੰਨ] ਇਆ ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *[jēkar_tūṃ] [mann] iā tāṃ tūṃ pachtā ēgā*
	Rule fired	>{PRON,ANIMT,CCLB,^CAG,^NOO:null:agt} {V,^VAUX,^AGTRES,^VOCURR,^COO:+AGTRES, +CCLB,+.@pred: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, *'agt'* relation is resolved between two analysis windows as shown below. agt(agree(icl>do).@not.@pred, you(icl>person))
Iteration:10	Node-net	[ਮੰਨ] [ਇਆ] ਤਾਂ ਤੂੰ ਪਛਤਾ ਏਗਾ *[mann] [iā] tāṃ tūṃ pachtā ēgā*
	Rule fired	+{V,m,sg:+IYA,+.@sg.@male:null}{[ਇਆ]: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 *'agree(icl>do)'* is modified as *'agree(icl>do).@not.@pred.@sg.@male'*.
Iteration:11	Node-net	[ਮੰਨ_ਇਆ] [ਤਾਂ] ਤੂੰ ਪਛਤਾ ਏਗਾ *[mann_ iā] tāṃ tūṃ pachtā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:12	Node-net	ਮੰਨ_ਇਆ [ਤਾਂ] [ਤੂੰ] ਪਛਤਾ ਏਗਾ *mann_ iā [tāṃ] [tūṃ] pachtā ēgā*
	Rule fired	-{CCL,^V,^N,^PRON,^ADJ,^ADV:null:null} {PRON:+CCL:null}
	Action taken	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.
Iteration:13	Node-net	[ਮੰਨ_ਇਆ] [ਤਾਂ_ਤੂੰ] ਪਛਤਾ ਏਗਾ *[mann_ iā] [tāṃ_ tūṃ] pachtā ēgā*
	Rule fired	No rule fired.
	Action	The analysis windows are shifted to right.

	taken	
Iteration:14	Node-net	ਮੰਨ_ਇਆ [ਤਾਂ_ਤੂੰ] [ਪਛਤਾ] ਏਗਾ *mann_ iā [tāṃ_ tūṃ] [pachtā] ēgā*
	Rule fired	>{PRON,ANIMT,CCL,^CAG,^NOO:null:agt} {V,^VAUX,^AGTRES,^VOCURR,^COO:+AGTRES, +CCL: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, *'agt'* relation is resolved between two analysis windows. agt(repent(icl>do), you(icl>person))
Iteration:15	Node-net	[ਮੰਨ_ਇਆ] [ਪਛਤਾ] ਏਗਾ *[mann_ iā] [pachtā] ēgā*
	Rule fired	>{V,CCLB:+.@entry:con}{V,CCL:null:null}
	Action taken	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. agt:01(repent(icl>do), you(icl<person)) con(agree(icl>do).@not.@pred.@sg.@male.@entry, :01)

Iteration:16	Node-net	[ਪਛਤਾ] [ਏਗਾ] *[pachtā] [ēgā]*
	Rule fired	+{V,m,sg:+.@future.@sg.@male:null}{[ਏਗਾ]: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 *'repent(icl>do)'* is modified as *'repent(icl>do).@future.@sg.@male'*.
Iteration:17	Node-net	[ਪਛਤਾ_ਏਗਾ] *[pachtā_ ēgā]*
	Rule fired	No rule fired.
	Action taken	Since, it is a last node so *'.@entry'* is added to UW *'repent(icl>do).@future.@sg.@male'* and UNL generation process completes at this iteration.

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).@not.@pred.@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.

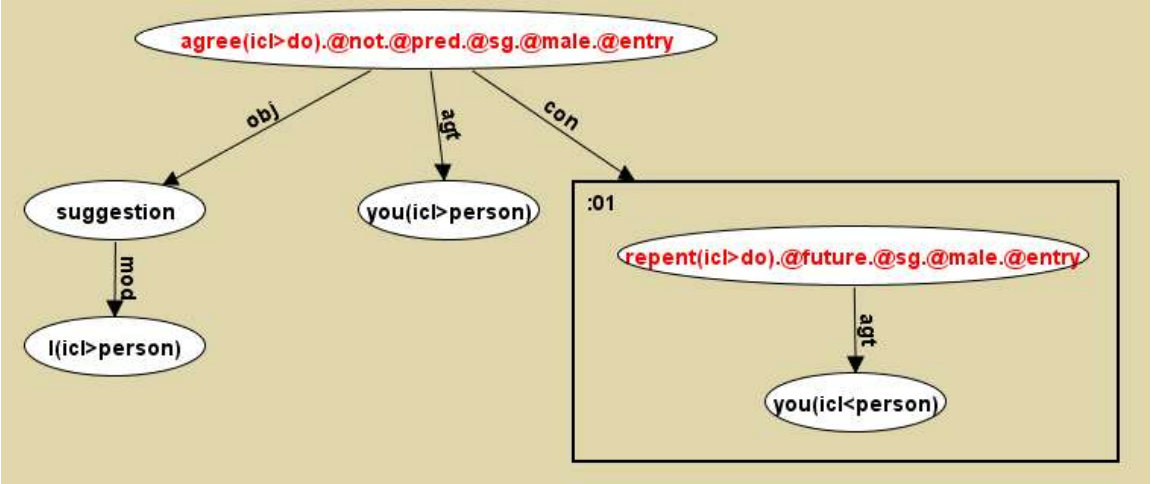

Figure 4.7: UNL graph for UNL expression given in (4.61)

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 ਜਿੱਦਾਂ *jiddāṃ* and ਓਦਾਂ *ōdāṃ* act as the clause delimiter for adverb clause for manner in Punjabi. The word ਜਿੱਦਾਂ *jiddāṃ* indicates the beginning of the manner condition and is represented with *'MCLB'* attribute (manner clause beginning), while word ਓਦਾਂ *ōdāṃ* 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: ਜਿੱਦਾਂ ਤੂੰ ਕਹੇਂਗਾ ਓਦਾਂ ਮੈਂ ਦੌੜਾਂਗਾ । …(4.62)

Transliterated Punjabi sentence: *jiddāṃ tūṃ kahēṅgā ōdāṃ maiṃ dauṛāṅgā.*

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)

Iteration:1	Node-net	[ਜਿੱਦਾਂ] [ਤੂੰ] ਕਹਿ ਏਗਾ ਓਦਾਂ ਮੈਂ ਦੌੜ ਏਗਾ *[jiddāṃ] [tūṃ] kahi ēgā ōdāṃ maiṃ dauṛ ēgā*
	Rule fired	-{MCLB,^V:null:null}{PRON,^MCLB:+MCLB:null}
	Action taken	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.
Iteration:2	Node-net	[ਜਿੱਦਾਂ_ਤੂੰ] [ਕਹਿ] ਏਗਾ ਓਦਾਂ ਮੈਂ ਦੌੜ ਏਗਾ *[jiddāṃ_tūṃ] [kahi] ēgā ōdāṃ maiṃ dauṛ ēgā*
	Rule fired	>{PRON,ANIMT,MCLB,^CAG,^NOO,^PSP,^POS,^KI, ^TCLB,^TCL,^CCLB,^CCL:null:agt} {V,^AGTRES,^VOCURR,^COO:+AGTRES,+MCLB :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, *'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

		for further processing. agt(say(icl>do), you(icl>person))
Iteration:3	Node-net	[ਕਹਿ] [ਏਗਾ] ਓਦਾਂ ਮੈਂ ਦੌੜ ਏਗਾ *[kahi] ēgā ōdāṃ maiṃ dauṛ ēgā*
	Rule fired	+{V,m,sg:+.@future.@sg.@male:null}{[ਏਗਾ]: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 *'say(icl>do)'* is modified as *'say(icl>do).@future.@sg.@male'*.
Iteration:4	Node-net	[ਕਹਿ_ਏਗਾ] [ਓਦਾਂ] ਮੈਂ ਦੌੜ ਏਗਾ *[kahi_ ēgā] [ōdāṃ] maiṃ dauṛ ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:5	Node-net	ਕਹਿ_ਏਗਾ [ਓਦਾਂ] [ਮੈਂ] ਦੌੜ ਏਗਾ *kahi_ ēgā [ōdāṃ] [maiṃ] dauṛ ēgā*
	Rule fired	-{MCL:null:null}{PRON,^MCL:+MCL:null}
	Action taken	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

		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	[ਕਹਿ_ਏਗਾ] [ਓਦਾਂ_ਮੈਂ] ਦੌੜ ਏਗਾ *[kahi_ ēgā] [ōdāṃ_maiṃ] dauṛ ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:7	Node-net	ਕਹਿ_ਏਗਾ [ਓਦਾਂ_ਮੈਂ] [ਦੌੜ] ਏਗਾ *kahi_ ēgā [ōdāṃ_maiṃ] [dauṛ] ēgā*
	Rule fired	>{PRON,ANIMT,MCL,^CAG,^NOO,^PSP,^POS,^KI, ^TCLB,^TCL,^CCLB,^CCL:null:agt} {V,^AGTRES,^VOCURR,^COO: +AGTRES,+MCL: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, *'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), I(icl>person))
Iteration:8	Node-net	[ਕਹਿ_ਏਗਾ] [ਦੌੜ] ਏਗਾ *[kahi_ ēgā] [dauṛ] ēgā*
	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 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) agt:01(run(icl>do).@future.@sg.@male.@entry, I(icl>person))
Iteration:9	Node-net	[ਦੌੜ] [ਏਗਾ] *[dauṛ] [ēgā]*
	Rule fired	+{V,m,sg:+.@future.@sg.@male:null}{[ਏਗਾ]: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 *'run(icl>do)'* is modified as *'run(icl>do).@future.@sg.@male'*.
Iteration:10	Node-net	[ਦੌੜ_ਏਗਾ] *[dauṛ_ ēgā]*
	Rule fired	No rule fired.
	Action taken	Since, it is a last node so *'.@entry'* is added to UW *'run(icl>do).@future.@sg.@male'* and UNL generation process completes at this iteration.

After processing of input example sentence given in (4.62), system generates equivalent UNL expression as given in (4.63).

{unl}

agt(say(icl>do).@future.@sg.@male.@entry, you(icl>person))

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

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

{/unl} …(4.63)

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

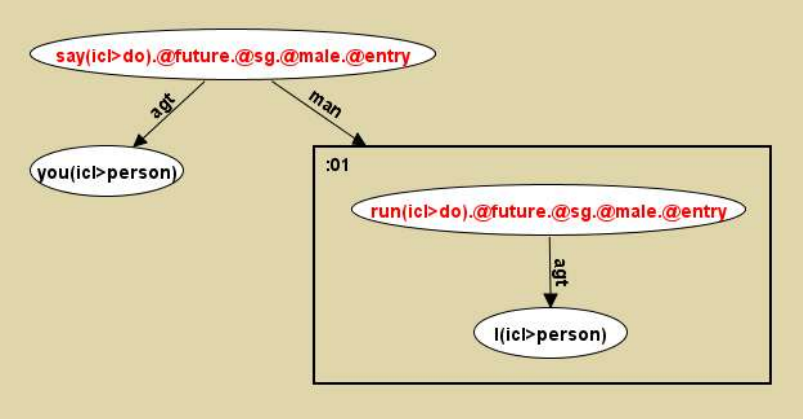

Figure 4.8: UNL graph for UNL expression given in (4.63)

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 ਜਿੱਥੇ *jitthē* and ਉੱਥੇ *utthē* act as the clause delimiters for adverb clause for place in Punjabi. The word ਜਿੱਥੇ *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: ਜਿੱਥੇ ਉਹ ਕਹੇਗਾ ਉੱਥੇ ਮੈਂ ਜਾਵਾਂਗਾ । …(4.64)

Transliterated Punjabi sentence: *jitthē uh kahēgā utthē maiṃ jāvāṅgā.*

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)

Iteration:1	Node-net	[ਜਿੱਥੇ] [ਉਹ] ਕਹਿ ਏਗਾ ਉੱਥੇ ਮੈਂ ਜਾ ਏਗਾ *[jitthē] [uh] kahi ēgā utthē maiṃ jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:2	Node-net	ਜਿੱਥੇ [ਉਹ] [ਕਹਿ] ਏਗਾ ਉੱਥੇ ਮੈਂ ਜਾ ਏਗਾ *Jitthē [uh] [kahi] ēgā utthē maiṃ jā ēgā*
	Rule fired	>{PRON,ANIMT,^CAG,^NOO,^PSP,^POS,^KI,^TCLB, ^TCL,^CCLB,^CCL:null:agt} {V,^AGTRES,^VOCURR,^COO:+AGTRES: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, *'agt'* relation is resolved between two analysis windows as shown below. agt(say(icl>do), he(icl>person))
Iteration:3	Node-net	[ਜਿੱਥੇ] [ਕਹਿ] ਏਗਾ ਉੱਥੇ ਮੈਂ ਜਾ ਏਗਾ *[jitthē] [kahi] ēgā utthē maiṃ jā ēgā*
	Rule fired	-{PCLB,^V:null:null}{V,^PCLB:+PCLB:null}
	Action taken	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.

		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.
Iteration:4	Node-net	[ਜਿੱਥੇ_ਕਹਿ] [ਏਗਾ] ਉੱਥੇ ਮੈਂ ਜਾ ਏਗਾ *[jitthē_khi] [ēgā] utthē maiṃ jā ēgā*
	Rule fired	+{V,m,sg:+.@future.@sg.@male:null}{[ਏਗਾ]: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 *'say(icl>do)'* is modified as *'say(icl>do).@future.@sg.@male'*.
Iteration:5	Node-net	[ਜਿੱਥੇ_ਕਹਿ_ਏਗਾ] [ਉੱਥੇ] ਮੈਂ ਜਾ ਏਗਾ *[jitthē_khi_ēgā] [utthē] maiṃ jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:6	Node-net	ਜਿੱਥੇ_ਕਹਿ_ਏਗਾ [ਉੱਥੇ] [ਮੈਂ] ਜਾ ਏਗਾ *jitthē_khi_ēgā [utthē] [maiṃ] jā ēgā*
	Rule fired	-{PCL:null:null}{PERPRON,^PCL:+PCL:null}
	Action taken	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 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.
Iteration:7	Node-net	[ਜਿੱਥੇ_ਕਹਿ_ਏਗਾ] [ਉੱਥੇ_ਮੈਂ] ਜਾ ਏਗਾ *[jitthē_khi_ēgā] [utthē_maiṃ] jā ēgā*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:8	Node-net	ਜਿੱਥੇ_ਕਹਿ_ਏਗਾ [ਉੱਥੇ_ਮੈਂ] [ਜਾ] ਏਗਾ *jitthē_khi_ēgā [utthē_maiṃ] [jā] ēgā*
	Rule fired	>{PERPRON,ANIMT,PCL,^CAG,^NOO,^PSP,^POS,^KI , ^TCLB,^TCL,^CCLB,^CCL:null:agt} {V,^AGTRES,^VOCURR,^COO:+AGTRES,+PCL: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, *'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>do), I(icl>person))
Iteration:9	Node-net	[ਜਿੱਥੇ_ਕਹਿ_ਏਗਾ] [ਜਾ] ਏਗਾ *[jitthē_khi_ēgā] [jā] ēgā*

	Rule fired	>{V,PCLB:+.@entry:null}{V,PCL:null:plc}
	Action taken	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>do)'* and it replaces UW *'go(icl>do)'* with ':01' as shown in resolved relation *'plc'*. Here, the system also converts earlier resolved UNL relations involving *'go(icl>do)'* as composite relation as shown below. plc(say(icl>do), :01) agt:01(go(icl>do), I(icl>person))
Iteration:10	Node-net	[ਜਾ] [ਏਗਾ] *[jā] [ēgā]*
	Rule fired	+{V,m,sg:+.@future.@sg.@male:null}{[ਏਗਾ]: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 *'go(icl>do)'* is modified as *'go(icl>do).@future.@sg.@male'*.
Iteration:11	Node-net	[ਜਾ_ਏਗਾ] *[jā_ēgā]*
	Rule fired	No rule fired.

	Action taken	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).

{unl}

agt(say(icl>do).@future.@sg.@male.@entry, he(icl>person))

agt:01(go(icl>do).@entry.@future.@sg.@male.@entry, I(icl>person))

plc(say(icl>do).@future.@sg.@male.@entry, :01)

{/unl} …(4.65)

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

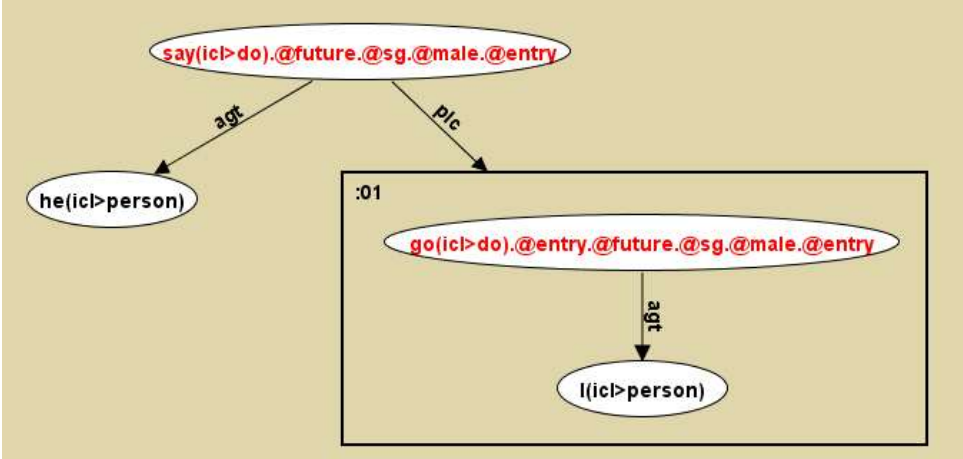

Figure 4.9: UNL graph for UNL expression given in (4.65)

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 *jō* and ਉਹ *uh* structure in adjective clause sentence. It means that it has a form of ਜੋ *jō* 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: ਜਿਸ ਬੱਚੇ ਨੂੰ ਰਾਮ ਨੇ ਬਚਾਇਆ ਉਹ ਸੋਹਣਾ ਸੀ। …(4.66)

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)

Iteration:1	Node-net	[ਜੋ] [ਬੱਚਾ] ਨੂੰ ਰਾਮ ਨੇ ਬਚਾ ਇਆ ਉਹ ਸੋਹਣਾ ਸੀ *[jō] [baccā] nūṃ rām nē bacā iā uh sōhṇā sī*
	Rule fired	-{ADJCLB,^V:null:null}{N,^ADJCLB:+ADJCLB:null}
	Action taken	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.
Iteration:2	Node-net	[ਜੋ_ਬੱਚਾ] [ਨੂੰ] ਰਾਮ ਨੇ ਬਚਾ ਇਆ ਉਹ ਸੋਹਣਾ ਸੀ *[jō_baccā] [nūṃ] rām nē bacā iā uh sōhṇā sī*
	Rule fired	+{N:+NOO:null}{[ੂੰ]:null :null}
	Action	It is a left composition rule and results into concatenation

	taken	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.
Iteration:3	Node-net	[ਜੋ_ਬੱਚਾ_ਨੂੰ] [ਰਾਮ] ਨੇ ਬਚਾ ਇਆ ਉਹ ਸੋਹਣਾ ਸੀ *[jō_baccā_nūṃ] [rām] nē bacā iā uh sōhṇā sī*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:4	Node-net	ਜੋ_ਬੱਚਾ_ਨੂੰ [ਰਾਮ] [ਨੇ] ਬਚਾ ਇਆ ਉਹ ਸੋਹਣਾ ਸੀ *Jō_baccā_nūṃ [rām] [nē] bacā iā uh sōhṇā sī*
	Rule fired	+{N,ANIMT:+CASE:null}{CASE:+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 *'CASE'* attribute in lexical semantic attribute list of node at left analysis window.
Iteration:5	Node-net	[ਜੋ_ਬੱਚਾ_ਨੂੰ] [ਰਾਮ_ਨੇ] ਬਚਾ ਇਆ ਉਹ ਸੋਹਣਾ ਸੀ *[jō_baccā_nūṃ] [rām_nē] bacā iā uh sōhṇā sī*
	Rule fired	No rule fired.
	Action taken	The analysis windows are shifted to right.
Iteration:6	Node-net	ਜੋ_ਬੱਚਾ_ਨੂੰ [ਰਾਮ_ਨੇ] [ਬਚਾ] ਇਆ ਉਹ ਸੋਹਣਾ ਸੀ *Jō_baccā_nūṃ [rām_nē] [bacā] iā uh sōhṇā sī*
	Rule fired	>{N,ANIMT,^CAG,^NOO,^PSP,^POS,^KI, ^TCLB,^TCL,^CCLB,^CCL:null:agt}

		{V,^AGTRES,^VOCURR,^COO:+AGTRES: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, *'agt'* relation is resolved between two analysis windows as shown below. agt(save(icl>do), Ram(icl>person))
Iteration:7	Node-net	[ਜੋ_ਬੱਚਾ_ਨੂੰ] [ਬਚਾ] ਇਆ ਉਹ ਸੋਹਣਾ ਸੀ *[jō_baccā_nūṃ] [bacā] iā uh sōhṇā sī*
	Rule fired	>{N,NOO,ADJCLB,^TIME:+.@entry:obj} {V:+OBJRES: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. 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. obj(save(icl>do), child(icl>person).@entry)
Iteration:8	Node-net	[ਬਚਾ] [ਇਆ] ਉਹ ਸੋਹਣਾ ਸੀ *[bacā] [iā] uh sōhṇā sī*
	Rule fired	+{V,m,sg:+IYA,+.@sg.@male:null}{[ਇਆ]: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, the attributes *'.@sg.@male'* is preceded by *'.@'* sign, so they are concatenated to corresponding UW as UNL attributes. Here, the UW *'save(icl>do)'* is modified

		as *‘save(icl>do).@sg.@male’*.
Iteration:9	Node-net	[ਬਚਾ_ਇਆ] [ਉਹ] ਸੋਹਣਾ ਸੀ *[bacā_ iā] [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	[ਬਚਾ_ਇਆ_ਉਹ] [ਸੋਹਣਾ] ਸੀ *[bacā_ iā_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 ਜੋ *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 ਜੋ structure 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))

		aoj(beautiful(icl>state).@entry, :01)
Iteration:11	Node-net	[ਸੋਹਣਾ] [ਸੀ] *[sōhṇā] [sī]*
	Rule fired	+{ADJ:+.@past:null}{[ਸੀ]: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	[ਸੋਹਣਾ_ਸੀ] *[sōhṇā_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} …(4.67)

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)'*.

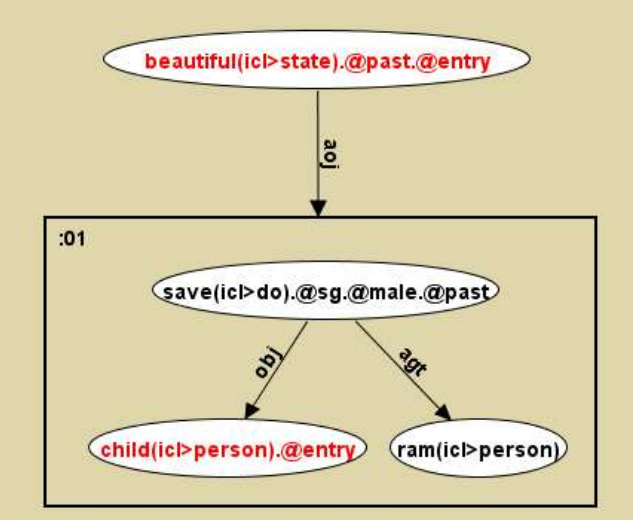

Figure 4.10: UNL graph for UNL expression given in (4.67)

The above mentioned strategies empower Punjabi EnConverter to successfully convert simple as well as complex Punjabi sentences into their equivalent UNL expressions.

Chapter Summary

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.

Chapter 5

UNL-Punjabi DeConverter

5.1 Introduction

The UNL-Punjabi DeConverter generates natural language Punjabi sentence from a given UNL expression. This generation process is based on the predicate-centric nature of the UNL. The DeConverter transforms an input UNL expression into the directed hyper-graph structure known as node-net. The root node of node-net is called as entry node and it acts as the main predicate. The traversing of node-net starts from entry node and system traverse the entire UNL graph to produce equivalent sentence in the target language. The DeConverter system makes an extensive use of rule base, including, morphological rules, syntax planning rules and function word insertion rules during this process (Vora, 2002). Since one generates natural language sentences during the process of DeConversion, we first discuss the issues involved in developing a Natural Language Generation (NLG) system.

5.2 Natural Language Generation system

NLG system is used to convert non-linguistic representation of information into a natural language understandable by human beings. There are six important activities that need to be carried out by an NLG system. These activities are content determination, discourse planning, sentence aggregation, lexicalization, referring expression generation and linguistic realization (Hurshikesh, 2002; Nalawade, 2007).

5.3 Punjabi DeConverter as an NLG system

Punjabi DeConverter performs the tasks of an NLG system in four intermediate steps, namely, lexicon selection, morphology generation of lexical words, function word insertion or case marking and syntax planning (Vachhani, 2006).

5.3.1 Lexeme selection

Punjabi DeConverter makes use of Punjabi-UW dictionary as the lexicon to select word entries during DeConversion process. A dictionary entry contains the correspondence

between a concept and a word; and information concerning the morphological, syntactic and semantic properties of a word.

5.3.2 Morphology generation of lexical words

Morphology involves the mapping of the words proposed in content determination stage to its more natural meaning. In this stage, words are changed (something is added or removed) to get proper sense of words in terms of gender, number, tense, aspect and modality.

5.3.3 Function word insertion or case marking

The relationship of noun phrase with its governing head is given by the function words or case markers of the language. It is used to insert function words like case marker or postpositions and conjunctions in Punjabi (*e.g.*, ਨੇ *nē*, ਨੂੰ *nūṃ*, ਉੱਤੇ *uttē 'over' etc.*) to the morphed words generated by morphology phase.

5.3.4 Syntax planning

The major objective of NLG system is to produce the sentences that are natural and as close as possible to what human beings would produce given a same domain and a similar need to communicate. It is evident that some word orders are considered more natural than others in a language. The syntax planning deals with the arrangements of words in the generated output so that output matches with the natural language sentence (Vachhani, 2006).

5.4 Architecture of UNL-Punjabi DeConverter

A general architecture of Punjabi DeConverter is given in Figure 5.1. It makes use of language-independent and language-dependent components during the generation process. Resources in the rectangular shape represent language-independent processes and that in oval shape represent language-dependent processes. First stage of DeConverter is UNL parser which parses the input UNL expression to build a node-net from the input UNL expression. The node-net has a Directed Acyclic Graph (DAG) structure. During lexeme selection stage, target language (*i.e.*, Punjabi) root words and their dictionary attributes are selected for the given UWs in the input UNL expression from the Punjabi-UW dictionary. After that, the nodes are ready for generation of morphology according to the target language in the morphology phase. In this stage, the root words may be changed or something can be added or removed to get the complete

sense of the words. The system makes use of morphology rules for this purpose. In function word insertion phase, the function words or case markers, such as ਨੇ *nē*, 'ਦੇ ਨਾਲ' *'dē nāl'*, ਨੂੰ *nūṃ*, ਤੋਂ *tōṃ*, ਦੇ ਲਈ *dē laī*, ਵਾਸਤੇ *vāsatē*, ਦਾ *dā*, ਦੇ *dē*, ਦੀ *dī etc.* are inserted to the morphed words. These function words are inserted in the generated sentence, based on the rule base designed for this purpose. Finally, the syntax planning phase is used to define the word order in the generated sentence so that output matches with a natural language sentence (Nalawade, 2007; Singh *et al.*, 2007).

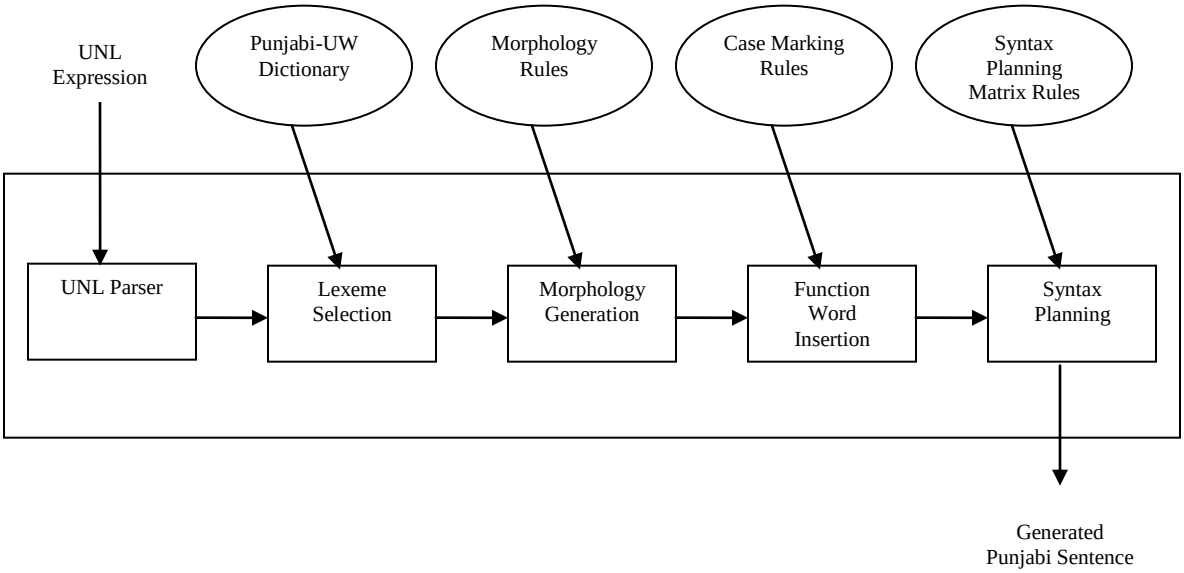

Figure 5.1: Architecture of UNL-Punjabi DeConverter

The working of Punjabi DeConverter is illustrated with the example sentence given in (5.1).

Punjabi sentence: ਮੁੰਡੇ ਨੇ ਬਾਗ਼ ਵਿਚ ਫ਼ੁੱਟਬਾਲ ਖੇਡਿਆ । …(5.1)

Transliterated sentence: *muṇḍē nē bāġ vic fuṭṭbāl khēḍiā.*

Equivalent English sentence: The boy played football in the garden.

The UNL expression for example sentence (5.1) is given in (5.2).

{unl}

agt(play(icl>do).@past.@entry, boy(icl>male person))

obj(play(icl>do).@past.@entry, football(icl>game))

plc(play(icl>do).@past.@entry, garden(icl>place))

{/unl} …(5.2)

In order to convert UNL expression given in (5.2) to natural language Punjabi sentence, Punjabi DeConverter is used. The UNL expression acts as input for Punjabi DeConverter. The UNL parser checks the input UNL expression for errors and generates the node-net or UNL graph as depicted in Figure 5.2.

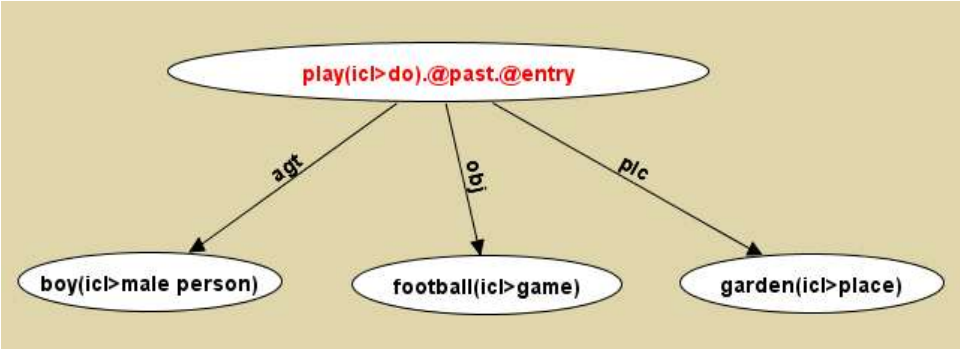

Figure 5.2: UNL graph generated by UNL parser for UNL expression given in (5.2)

The lexeme selection phase populates the node-list with equivalent Punjabi words for the UWs given in the input UNL expression. The populated node-list is given in (5.3).

Node$_1$: Punjabi word: ਖੇਡ *khēḍ;* UW: play(icl>do).@past.@entry

Node$_2$: Punjabi word: ਮੁੰਡਾ *muṇḍā*; UW: boy(icl>male person)

Node$_3$: Punjabi word: ਫੁੱਟਬਾਲ *fuṭṭbāl*; UW: football(icl>game)

Node$_4$: Punjabi word:ਬਾਗ *bāġ*; UW: garden(icl>place) …(5.3)

In morphology phase, morphological rules are applied to modify Punjabi words stored in the nodes according to UNL attributes given in input UNL expression and dictionary attributes retrieved from Punjabi-UW dictionary. The nodes given in (5.3) are processed by morphology rules. The processed nodes are given in (5.4).

Node$_1$: Punjabi word: ਖੇਡਿਆ *khēḍiā*; UW: play(icl>do).@past.@entry

Node$_2$: Punjabi word: ਮੁੰਡੇ *muṇḍē*; UW: boy(icl>male person)

Node$_3$: Punjabi word: ਫੁੱਟਬਾਲ *fuṭṭbāl*; UW: football(icl>game)

Node$_4$: Punjabi word:ਬਾਗ *bāġ*; UW: garden(icl>place) …(5.4)

It is evident from nodes given in (5.4) that, in morphology phase ਖੇਡ *khēḍ 'play'* is changed to ਖੇਡਿਆ *khēḍiā 'played'* and ਮੁੰਡਾ *muṇḍā 'boy'* is changed to ਮੁੰਡੇ *muṇḍē 'boy'* by morphology rules. The function word insertion phase inserts the function words in the

morphed lexicon. The nodes processed by function word insertion phase are given in (5.5).

$Node_1$: Punjabi word: ਖੇਡਿਆ *khēḍiā*; UW: play(icl>do).@past.@entry

$Node_2$: Punjabi word: ਮੁੰਡੇ ਨੇ *muṇḍē nē*; UW: boy(icl>male person)

$Node_3$: Punjabi word: ਫ਼ੁੱਟਬਾਲ *fuṭṭbāl*; UW: football(icl>game)

$Node_4$: Punjabi word:ਬਾਗ਼ ਵਿਚ *bāġ vic*; UW: garden(icl>place) …(5.5)

In this phase, case markers ਨੇ *nē* and ਵਿਚ *vic 'in'* are added to $Node_2$ and $Node_4$, respectively, according to the function word insertion rule base. The syntax planning phase traverses the nodes given in (5.5) in a specific sequence based on the syntax planning rule base for Punjabi language. The sequence for processing of nodes is given in (5.6) and Punjabi sentence generated by this sequence is given in (5.7).

$Node_2$ $Node_4$ $Node_3$ $Node_1$ …(5.6)

ਮੁੰਡੇ ਨੇ ਬਾਗ਼ ਵਿਚ ਫ਼ੁੱਟਬਾਲ ਖੇਡਿਆ । …(5.7)

muṇḍē nē bāġ vic fuṭṭbāl khēḍiā.

It is evident from generated Punjabi sentence given in (5.7) that system is able to convert the input UNL expression to Punjabi successfully.

The description and implementation details of each phase of Punjabi DeConverter is given in next sections.

5.5 UNL parser

UNL parser is first phase of UNL-Punjabi DeConverter. It is used to parse the input UNL expression to report the errors if any in the input expression. If input expression is in proper format or free from errors, then it builds semantic net known as node-net structure for input UNL expression, otherwise it reports the errors. This node-net is commonly called as UNL graph. The UNL graph consists of nodes and edges. A node in UNL graph represents a concept in the form of UW. An edge in node-net represents a UNL binary relation between two nodes. The edges in a UNL graph are directed from the parent node to child node (Singh, 2007). For a UNL binary relation *rel*(UW1, UW2), UW1 is considered as parent of UW2. Thus, in a UNL graph the edge will be directed from UW1 to UW2 as shown in Figure 5.3. The UNL relation between UWs is used to label the edges between the nodes in the UNL graph, as shown in Figure 5.3.

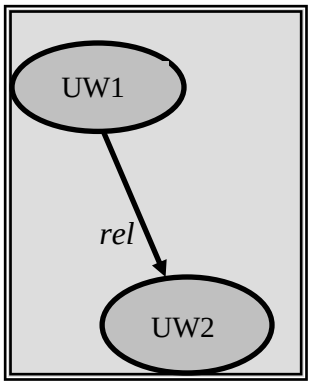

Figure 5.3: A UNL graph representation for UNL binary relation *rel*(UW1, UW2)

The root node of UNL graph is called as entry node and it acts as the main predicate. The main predicate is detected by presence of *'@entry'* attribute with the UW given in input UNL expression. As discussed earlier, the traversing of node-net starts from entry node and the system traverse the entire UNL graph to produce the equivalent sentence in the target language. Sometimes, during the traversal of UNL graph, the system may encounter a stage where a node has two or more parents as shown in Figure 5.4. Here, node 'B' has two parents 'D' and 'A'. For a node having more than one parent, the parents need to be traversed before that node. To accomplish this, system has to maintain the access path from child to its parent (Singh, 2007). The back edges from every child node to its parent node are maintained for this purpose, *i.e.*, for a UNL binary relation *rel*(UW_1, UW_2), the back edge directed from UW2 to UW1 is also maintained as shown in Figure 5.5.

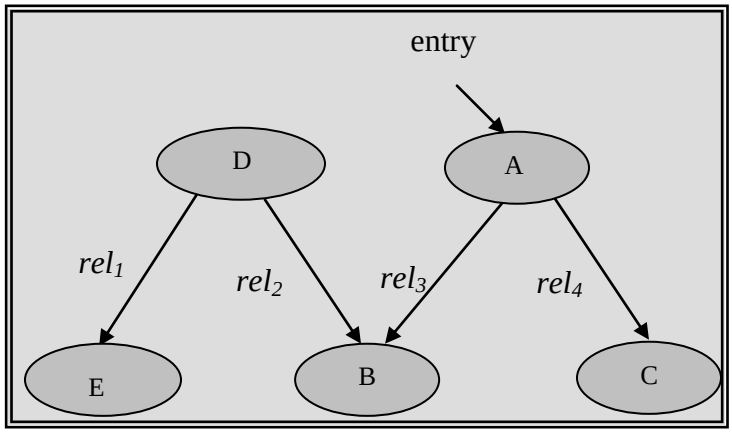

Figure 5.4: UNL graph with a node having two parents

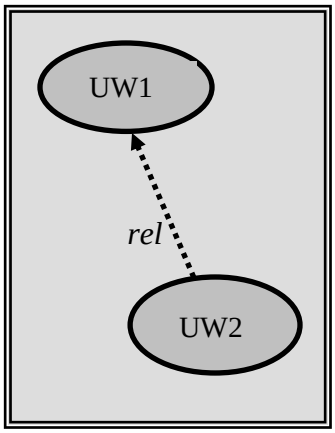

Figure 5.5: Back edge in UNL graph for *rel*(UW1, UW2)

The back edges are introduced in UNL graph given in Figure 5.4 to handle the traversing of a node having multiple parents. This is illustrated in Figure 5.6.

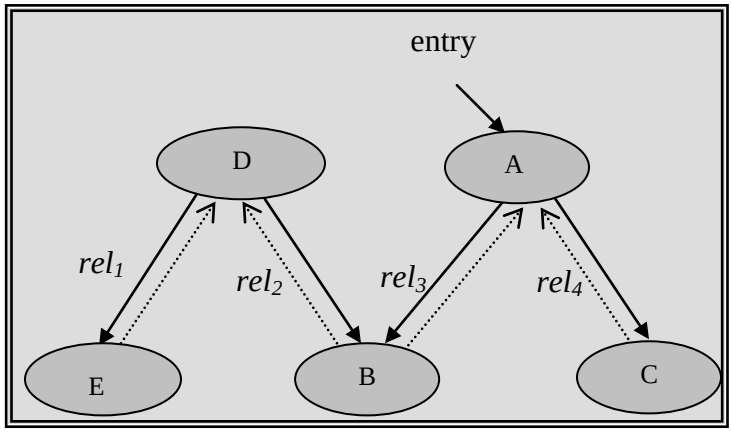

Figure 5.6: UNL graph with back edges to handle a node having two parents

In UNL expression, compound UWs are used to represent compound concepts. This concept of compound UWs is illustrated below with the help of an example sentence given in (5.8).

The crop was cultivated 12 days back. …(5.8)

The UNL expression of example sentence (5.8) is given in (5.9).

{unl}
obj(cultivate.@entry, crop)
tim(cultivate.@entry, :01)
man:01(back, day)
qua:01(day, 12)
{/unl} …(5.9)

The UNL graph involving a scope node for UNL expression given in (5.9) is depicted in Figure 5.7.

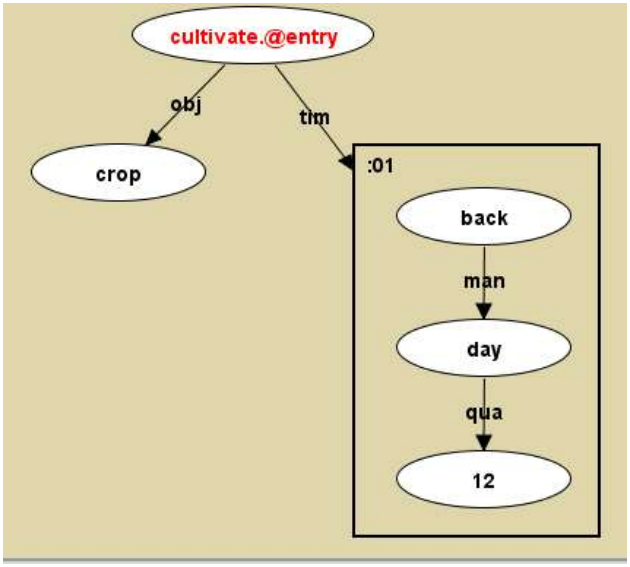

Figure 5.7: UNL graph with scope node for UNL expression given in (5.9)

In this phase, one needs to develop data structures that will subsequently be used in next phases of UNL-Punjabi DeConverter. The procedure that has been used to create these data structures is given in PseudoCode 5.1.

PseudoCode 5.1: Building data structures in UNL parser

begin

for (every relation in input UNL Expression)

Node1 = UW1;

Node2 = UW2;

Relation = Relation between UW1 and UW2;

if (Node1 not in NodeList)

add Node1 to NodeList;

end-if

if (Node2 not in NodeList)

add Node2 to NodeList;

end-if

add relation to RelationList;

if (UW1 contains *@entry* attribute)

add UW1 and their Scope in ScopeList;

else

if (UW2 contains *@entry* attribute)

add UW2 and their Scope in ScopeList;

end-if

end-if

end-for

end

Here, Node1 and Node2 are two nodes of UNL graph, Relation represents relation between nodes, Scope represents an element of the ScopeList, NodeList is a linked list of Nodes, RelationList is a linked list of Relations and ScopeList is a linked list of Scope elements.

5.6 Lexeme selection

Lexeme selection is the process of selecting target language words for the UWs given in input UNL expression. During lexeme selection, UWs are searched in the dictionary along with constraints specified in input UNL expression. This phase uses Punjabi-UW dictionary for this task. This dictionary contains Punjabi root word as the headword, a UW and a set of morphological, syntactic and semantic attributes in an entry. The attributes are used to define the behavior of word or a morpheme in a sentence. The UW uses restrictions or constraint list to represent the concepts unambiguously, that helps to achieve Word Sense Disambiguation (WSD) during the generation process (Nalawade, 2007).

For example, during the processing of UNL expression given in (5.2) in this phase, system searches each UW given in input UNL expression, *i.e.*, *'play(icl>do)'*, *'boy(icl>male person)'*, *'football(icl>game)'* and *'garden(icl>place)'* from the Punjabi-UW dictionary. It retrieves the Punjabi headwords, namely, ਖੇਡ *khēḍ 'play'*, ਮੁੰਡਾ *muṇḍā 'boy'*, ਫੁੱਟਬਾਲ *fuṭṭbāl 'football'* and ਬਾਗ਼ *bāġ 'garden'*, respectively, from the lexicon for these UWs.

5.7 Morphology generation

In this phase, headwords are modified according to the morphology of the target language. The system makes use of generation rules during this process. These generation rules are designed on the basis of analysis of Punjabi morphology carried

out for this purpose. There are three categories of morphology that have been identified for the purpose of conversion of UNL expression to equivalent Punjabi language sentences (Vachhani, 2006). These are:

- Attribute label resolution morphology
- Relation label resolution morphology and
- Noun, adjective, pronoun and verb morphology.

5.7.1 Attribute label resolution morphology

Attribute label morphology deals with the generation of Punjabi words on the basis of UNL attributes attached to a node and its grammatical attributes retrieved from lexicon. The root words retrieved from Punjabi-UW dictionary are changed in this phase depending on their Gender, Number, Person, Tense, Aspect, Modality (GNPTAM) and vowel ending information. Verbs play a vital role in extracting this information from a given sentence (Singh *et al.*, 2007). It provides the gender information, namely, masculine, feminine or non-specific; number information, namely, singular, plural or non-specific; person information, namely, first, second or third; tense information, namely, past, present or future, about a given word. It also provides the information about the aspect, *i.e.*, perfective, completive, frequentative, habitual *etc.*

The attribute label morphology also depends upon the vowel ending of verbs. Punjabi has ten vowels, represented as ਾ (ਆ *ā*), ਿ (ਇ *i*), ੀ (ਈ *ī*), ੁ (ਉ *u*), ੂ (ਊ *ū*), ੇ (ਏ *ē*), ੈ (ਐ *ai*), ੋ (ਓ *ō*), ੌ (ਔ *au*) and ਮੁਕਤਾ *muktā* (ਅ *a*) which has no sign. Vowels other than ਅ *a* (ਮੁਕਤਾ) *muktā* are represented by accessory signs written around (*i.e.*, below, above, to the right or to the left) the consonant signs, popularly known as signs for *matras* (Bahri and Walia, 2003). The Punjabi language attribute label morphology for masculine gender is given in Appendix-B.

The attribute label morphology also deals with the generations of articles in the target language. For example, definite articles (typically arise from demonstratives meaning '*that')* are represented in UNL expression by *'@def'* attribute and it results the generation of word ਉਹ *uh 'that'* in Punjabi. Similarly, indefinite articles (typically arise from adjectives meaning *'one'*) are represented by *'@indef'* attribute and this results into the generation of Punjabi word ਇੱਕ *ikk* or

nothing depending upon the number of the words it qualifies in the attribute label morphology (Jain, 2005).

5.7.2 Relation label resolution morphology

The relation label morphology manages the prepositions in English or postpositions in Punjabi, because prepositions in English are similar to postpositions in Punjabi. These link noun, pronoun, and phrases to other parts of the sentence. Some Punjabi postpositions are ਨੇ *nē*, ਨੂੰ *nūṃ*, ਉੱਤੇ *uttē 'over'*, ਦਾ *dā 'of'*, ਕੋਲੋਂ *kōlōṃ 'from'*, ਨੇੜੇ *nēṛaē 'near'*, ਲਾਗੇ *lāgē 'near' etc.* In Punjabi, postpositions follow the noun or pronoun unlike English, where these precede the noun or pronoun, and thus termed prepositions (Gill, 2008). Insertion of these words in the generated output depends upon the information encoded in the UNL relations of a given UNL expression. In relation label morphology, most UNL relation labels introduce postpositions (also known as function words or case markers) between child and the parent node during the generation process. The generation of these words depends upon UNL relation and the conditions imposed on parent and child nodes' attributes of the UNL relation. For the generation of these words a rule base has been prepared. Let us illustrate this concept with an example English sentence given in (5.10) and its equivalent Punjabi sentence given in (5.11). The UNL expression for this example sentence is given in (5.12).

The boy translated the sentence from English to Punjabi. …(5.10)

ਮੁੰਡੇ ਨੇ ਅੰਗਰੇਜ਼ੀ ਤੋਂ ਪੰਜਾਬੀ ਵਿਚ ਵਾਕ ਦਾ ਅਨੁਵਾਦ ਕੀਤਾ । …(5.11)

muṇḍē nē aṅgrēzī tōṃ pañjābī vic vāk dā anuvād kītā.

{unl}

agt(translate(icl>do).@past.@entry, boy(icl>male child))

src(translate(icl>do).@past.@entry, english(icl>language))

gol(translate(icl>do).@past.@entry, punjabi(icl>language))

obj(translate(icl>do).@past.@entry, sentence)

{/unl} …(5.12)

Here, the case markers ਨੇ *nē*, ਤੋਂ *tōṃ 'from'*, ਵਿਚ *vic 'to'* and ਦਾ *dā* are inserted in the morphed words due to the presence of UNL relations '*agt'*, '*src'*, '*gol'* and *'obj'*, respectively, in the UNL expression given in (5.12). The function word insertion phase of

UNL DeConverter, deals with the relation label resolution morphology (Hrushikesh, 2002; Jain, 2005; Singh *et al.*, 2007). The detailed discussion on function word insertion phase has been provided in Section 5.8.

5.7.3 Noun, adjective, pronoun and verb morphology

With attribute and relation label morphology, the system is able to generate the sentence very close to its natural form. The phonetic properties of the language are handled by the noun, adjective, pronoun and verb morphology of the DeConverter.

5.7.3.1 Noun morphology

Noun morphology deals with the properties of nouns to identify their behavior in the generation process. The nouns are analyzed on the basis of Gender, Number, Person, Case (GNPC) and their paradigm information. Punjabi noun paradigms are identified on the basis of their vowel ending. The nouns are affected by direct case (the case in which subject is directly referred) and oblique case (the case in which subject is not directly referred). Here, a part of the word is removed from the end and a new phoneme is added to the end of the word during the generation process. There is a group of nouns that change their form during the generation process while others retain their form and do not change in the generation form. In order to distinguish such nouns, the lexical attribute *'NOTCH'* (not changeable form) is used in the lexicon, *i.e.*, a noun that remains unchanged, and does not inflect for number or case. This is illustrated with Punjabi example sentence given in (5.13) and equivalent English sentence given in (5.14).

ਇਹ ਮਾਲੀ ਸਾਲ ਭਾਰਤ ਲਈ ਚੰਗਾ ਹੈ । …(5.13)

ih mālī sāl bhārat laī caṅgā hai.

This financial year is good for India. ….(5.14)

Here, the noun 'ਮਾਲੀ ਸਾਲ' *'mālī sāl' 'financial year'* has *'Na'* as the vowel ending attribute and it retains original form without inflecting for number and case, so it is stored in the lexicon with *'NOTCH'* attribute as given in (5.15).

[ਮਾਲੀ ਸਾਲ] {} "financial year(icl>fiscal year)" (N,NOTCH,MALE,INANI,Na)

<P,0,0>;/(*mālī sāl*) …(5.15)

The analysis of Punjabi nouns on the basis of their vowel ending, paradigms and GNPC information is given in Table 5.1.

Table 5.1: Noun morphology for Punjabi DeConverter

Gender	Vowel ending of noun/ Paradigm	Example	Suffix (Direct/ Oblique)	Number (Singular SG/ Plural PL)	Morph-ology	Example sentence
Male	ਮੁਕਤਾ *muktā* (ਅ *a*)	ਮੇਜ਼ *mēz 'table'*, ਫੁੱਲ *phull 'flower'*, ਹੱਥ *hatth 'hand' etc.*	Direct	SG	No Change	ਫੁੱਲ ਨਾ ਤੋੜੋ । *phull nā tōṛō.*
			Direct	PL	No Change	ਫੁੱਲ ਨਾ ਤੋੜੋ । *Phull nā tōṛō.*
			Oblique	SG	No Change	ਫੁੱਲ ਨੂੰ ਨਾ ਤੋੜੋ । *phull nūṃ nā tōṛō.*
			Oblique	PL	Delete nothing at the end of HW and insert ◌ਾਂ *āṃ*	ਫੁੱਲਾਂ ਨੂੰ ਨਾ ਤੋੜੋ । *phullāṃ nūṃ nā tōṛō.*
Male	◌ਾ (ਆ *ā*)	ਕਾਕਾ *kākā 'child'*, ਪੱਤਾ *pattā 'leaf'*, ਮੁੰਡਾ *muṇḍā 'boy' etc.*	Direct	SG	No Change	ਕਾਕਾ ਰੋਟੀ ਖਾਂਦਾ ਹੈ । *kākā rōṭī khāndā hai.*

			Direct	PL	Delete ਾ (ਆ *ā*) at the end of HW and insert ੇ (ਏ *ē*)	ਕਾਕੇ ਰੋਟੀ ਖਾਂਦੇ ਹਨ। *kākē rōṭī khāndē han.*
			Oblique	SG	Delete ਾ (ਆ *ā*) at the end of HW and insert ੇ (ਏ *ē*)	ਕਾਕੇ ਨੇ ਰੋਟੀ ਖਾਧੀ ਹੈ। *kākē nē rōṭī khādhī hai.*
			Oblique	PL	Delete ਾ (ਆ *ā*) at the end of HW and insert ਿਆਂ *iāṃ*	ਕਾਕਿਆਂ ਨੇ ਰੋਟੀ ਖਾਧੀ ਹੈ। *kākiāṃ nē rōṭī khādhī hai.*
Male	ੀ (ਈ *ī*), ੂ (ਊ *ū*)	ਧੋਬੀ *dhōbī 'washerman', ਗੁਰੂ gurū 'guru' etc.*	Direct	SG	No Change	ਧੋਬੀ ਰੋਟੀ ਖਾਂਦਾ ਹੈ। *dhōbī rōṭī khāndā hai.*
			Direct	PL	No Change	ਧੋਬੀ ਰੋਟੀ ਖਾਂਦੇ ਹਨ।

						dhōbī rōṭī khāndē han.
			Oblique	SG	No Change	ਧੋਬੀ ਨੇ ਰੋਟੀ ਖਾਧੀ ਹੈ। *dhōbī nē rōṭī khādhī hai.*
			Oblique	PL	Delete nothing at the end of HW and insert ਆਂ *āṃ*	ਧੋਬੀਆਂ ਨੇ ਰੋਟੀ ਖਾਧੀ ਹੈ। *dhōbīāṃ nē rōṭī khādhī hai.*
Female	ਮੁਕਤਾ *muktā* (ਅ *a*)	ਕਿਤਾਬ *kitāb 'book'*, ਲੱਤ *latt 'leg'*, ਬਾਂਹ *bāṃh 'arm' etc.*	-	SG	No Change	ਤੁਸੀਂ ਕਿਤਾਬ ਪੜ੍ਹੋ। *tusīṃ kitāb paṛhō.*
			-	PL	Delete nothing at the end of HW and insert ◌ਾਂ *āṃ*	ਤੁਸੀਂ ਕਿਤਾਬਾਂ ਪੜ੍ਹੋ। *tusīṃ kitābāṃ paṛhō*
Female	◌ਾ (ਆ *ā*)	ਸਭਾ *sabhā 'assembly'*, ਹਵਾ *havā 'air'*, ਕਵਿਤਾ	-	SG	No Change	ਹਵਾ ਗਰਮ ਹੈ। *havā garam hai.*

		kavitā ‘poem’ etc.				
				PL	Delete nothing at the end of HW and insert ਵਾਂ *vāṃ*	ਹਵਾਵਾਂ ਗਰਮ ਹਨ । *havāvāṃ garam han.*
Female	◌ਾਂ *āṃ*	ਮਾਂ *māṃ ‘mother’*, ਛਾਂ *chāṃ ‘shadow’ etc.*	-	SG	No Change	ਮਾਂ ਘਰ ਗਈ । *māṃ ghar gaī.*
				PL	Delete nothing at the end of HW and insert ਵਾਂ *vāṃ*	ਮਾਂਵਾਂ ਘਰ ਗਈਆਂ । *māṃvāṃ ghar gaīāṃ.*
Female	◌ੂ (ਊ *ū*), ◌ੀ (ਈ *ī*)	ਵਸਤੂ *vasatū ‘article’*, ਨਦੀ *nadī ‘river’ etc.*	-	SG	No Change	ਨਦੀ ਵਿਚ ਪਾਣੀ ਹੈ । *Nadī vic pāṇī hai.*
			-	PL	Delete nothing at the end of HW and	ਨਦੀਆਂ ਵਿਚ ਪਾਣੀ ਹੈ । *nadīāṃ vic pāṇī hai.*

					insert ਆਂ *āṃ*	
Female	ੁੰ *uṃ*, ੂੰ *ūṃ*	ਸਹੁੰ *sahuṃ* *'oath'*, ਜੂੰ *jūṃ* *'lice'* *etc.*	-	SG	No Change	ਉਸ ਨੇ ਜੂੰ ਮਾਰੀ । *us nē jūṃ mārī.*
			-	PL	Delete nothing at the end of HW and insert ਆਂ *āṃ*	ਉਸ ਦੇ ਸਿਰ ਵਿਚ ਜੂੰਆਂ ਹਨ । *us dē sir vic jūṃāṃ han.*

Implementation of noun morphology in UNL

In a UNL expression, information about plural number is represented with *'@pl'* attribute. Absence of *'@pl'* attribute conveys that the number being used is singular. The case, direct or oblique, is identified using the relation that a noun has with a verb or with another noun in the sentence (Singh *et al.*, 2007). The gender and vowel endings information are stored in the lexicon. The morphological rules based on word paradigms generate a noun form using lexical, relational and UNL attributes information. Morphology rules for Punjabi language have been designed on the basis of their GNPC and paradigm information. The format used for noun morphology rules is given in (5.16).

Ultimate deletion, Ultimate insertion, @Attrubute1, @Attrubute2, … ...(5.16)

Here, '*Ultimate deletion'* and '*Ultimate insertion'* represents character to be deleted or to be inserted at the end of headword and *'@Attrubute1, @Attrubute2'* and so on indicates the condition for the firing of the corresponding rule, *i.e.*, morphology rule will be applied if attributes present in the rule base matches with attributes of current UW in the

UNL expression and its dictionary attributes. The noun morphology rule base is illustrated with an example rule base entry, given in (5.17).

null, ◌ਾਂ, @pl,@FEMALE,@Na …(5.17)

Here, '*@pl,@FEMALE,@Na*' is the list of attributes to be matched for the firing of this rule, *i.e.*, if UW represents a noun having vowel ending ਮੁਕਤਾ *muktā* (ਅ) *a* (*i.e.* '*Na*') with feminine gender and plural number, then corresponding HW changes its original form by deleting nothing from the end of HW and by inserting ◌ਾਂ *āṃ* at the end of headword. Let us consider a UNL expression involving UW with attributes as '*book(icl>publication).@pl*'. The UW *'book(icl>publication)'* has ਕਿਤਾਬ *kitāb 'book'* as headword with '*Na*' and *'FEMALE'* as its dictionary attribute in Punjabi-UW lexicon. Here, all the conditions for the firing of rule given in (5.17) are satisfied, and it will result into insertion of '◌ਾਂ' *āṃ* at the end of headword ਕਿਤਾਬ *'book'* to form ਕਿਤਾਬਾਂ *kitābāṃ* '*books'* as the morphed word for the given attributes.

The rule base for generation of noun morphology is given in Table 5.2. These rules here are listed according to their priority of usage.

Table 5.2: Generation of noun morphology rules

S.No	Noun morphology rule
1.	null,,@past,@sg,@MALE,@NA,@INANI
2.	◌ਾ, ◌ੇ,@MALE,@sg,@past,@NA
3.	null,,@past,@progress,@sg,@MALE,@NA
4.	null,,@present,@progress,@sg,@MALE,@NA
5.	null,,@progress,@future,@sg,@MALE,@NA
6.	◌ਾ, ◌ੇ,@MALE,@sg,@progress,@NA
7.	◌ਾ, ◌ੇ,@MALE,@sg,@past,@NA
8.	◌ਾ, ◌ੇ,@MALE,@pl,@past,@NA,@progress,@present,@future
9.	null,,@MALE,@sg,@past,@NA,@progress
10.	null,,@present,@sg,@MALE,@NA
11.	◌ਾ,◌ੇ,@present,@sg,@MALE,@NA,@link
12.	◌ਾ,◌ੇ,@future,@sg,@MALE,@NA,@link

13.	◌ർ,◌ൈ,@present,@pl,@MALE,@NA,@sg
14.	◌ർ, ◌ൈ,@MALE,@sg,@past,@complete,@NA
15.	◌ർ, ◌ൈ,@MALE,@sg,@present,@obligation,@NA
16.	◌ർ, ◌ൈ,@MALE,@sg,@past,@obligation,@NA
17.	null,,@future,@sg,@MALE,@NA
18.	◌ർ, ◌ൈ,@progress,@future,@sg,@MALE,@NA
19.	null,,@MALE,@sg,@NI
20.	null,,@past,@custom,@sg,@MALE,@NA
21.	null,,@future,@sg,@MALE,@NA
22.	null,,@present,@custom,@sg,@MALE,@NA
23.	null,,@past,@repeat,@sg,@MALE,@NA
24.	◌ർ,◌ൈ,@past,@custom,@pl,@MALE,@NA
25.	null,,@past,@present,@female,@Na
26.	null,◌ർ◌ൊ,@past,@present,@pl,@female,@Na
27.	null, ◌ൈ,@indef
28.	null,, @NI,@FEMALE
29.	null,ങ്ങൾ,@pl ,@NI,@FEMALE
30.	null,◌ൈ,@progress,@pl
31.	null,◌ൈ,@complete,@future,@pl
32.	null, ◌ർ,@permission,@sg,@MALE
33.	null,◌ീ, @FEMALE,@sg
34.	null,◌ൈ,@ability,@present,@pl,@MALE
35.	◌ർ,◌ൊങ്ങൾ,@MALE,@pl,@NA,@future,@custom,@past,@present

An algorithm has also been developed to implement the noun morphology. This algorithm 5.1 is presented below.

Algorithm 5.1: Processing of noun morphology rule base

(i) Obtain the UW, number and case information for current noun from UNL expression and store this information into an attribute list. If number information is not present, consider the number as singular.

(ii) Obtain HW, gender and vowel ending information corresponding to UW from Punjabi-UW dictionary and append it to the attribute list.

(iii) Search the noun morphology rule base and fire the rule that has maximum similarity with the elements of attribute list. If two or more rules have the same similarity, the rule that appear first is fired.

5.7.3.2 Adjective morphology

Adjective morphology is used to generate proper form of adjectives in a sentence. Adjective morphology depends on the gender, number and suffix information of the head noun. It has been observed that some adjectives do not change their form with respect to head noun and remain unchanged, for example, ਲਾਲ *lāl 'red'*, ਸਾਫ਼ *sāf 'clean' etc.* There is another category of adjectives that changes their form depending upon the gender, number and suffix information of the head noun. For example, ਚਮਕੀਲਾ *camkīlā 'bright'*, ਚੰਗਾ *caṅgā 'good' etc.* changes to ਚਮਕੀਲੀ *camkīlī 'bright'*, ਚੰਗੀ *caṅgī 'good'*, respectively, in case of feminine gender. The adjectives that change their form are represented with *'AdjA'* lexical attribute in the lexicon as shown in an example adjective entry of Punjabi-UW dictionary given in (5.18).

[ਚੰਗ] {} "good(aoj>human)" (ADJ,AdjA) <P,0,0>;/(*caṅg*) …(5.18)

The adjective morphology for Punjabi language used in this work is given in Table 5.3.

Table 5.3: Adjective morphology

Gender	Example	Condition	Morphology	Example sentence
Male	ਚੰਗਾ *caṅgā 'good'*, ਚਿੱਟਾ *ciṭṭā 'white'* ਉੱਚਾ *uccā 'high'*, ਕਾਲ਼ਾ *kāḷā 'black' etc.* These words are stored in Punjabi-UW dictionary as	Singular	Insert ਾ (ਆ *ā*) at the end of HW	ਇਹ ਮੁੰਡਾ ਚੰਗਾ ਹੈ । *ih muṇḍā caṅgā hai.*

	ਚੰਗ *caṅg*, ਚਿੱਟ *ciṭṭ*, ਉੱਚ *ucc*, ਕਾਲ਼ *kāḷ* respectively.			
		Plural	Insert ੇ (ਏ *ē*) at the end of HW	ਇਹ ਮੁੰਡੇ ਚੰਗੇ ਹਨ। *ih muṇḍē caṅgē han.*
		Respect	Insert ੇ (ਏ *ē*) at the end of HW	ਤੁਸੀਂ ਚੰਗੇ ਹੋ। *tusīṃ caṅgē hō.*
Female		Singular	Insert ੀ (ਈ *ī*) at the end of HW	ਇਹ ਕੁੜੀ ਚੰਗੀ ਹੈ। *ih kuṛī caṅgī hai.*
		Plural	Insert ੀਆਂ *īāṃ* at the end of HW ੀਆਂ	ਇਹ ਕੁੜੀਆਂ ਚੰਗੀਆਂ ਹਨ। *ih kuṛīāṃ caṅgīāṃ han.*
		Respect	Insert ੇ (ਏ *ē*) at the end of HW	ਤੁਸੀਂ ਚੰਗੇ ਹੋ। *tusīṃ caṅgē hō.*

Implementation of adjective morphology

It has been observed that, unlike nouns, gender and number information of adjectives are not directly embedded in UNL expression. Most of the adjectives exhibit concordance with their head nouns and their heads are identified using relation label in UNL expression (Singh *et al.*, 2007). The rule format for adjective morphology is given in (5.19).

Ultimate insertion, @Attrubute1, @Attrubute2, … ...(5.19)

Here, '*Ultimate insertion'* represents character to be inserted at the end of headword and *'@Attrubute1, @Attrubute2'* and so on, indicate the condition for the firing of the corresponding rule, *i.e.*, morphology rule will be applied if attributes present in the rule base matches with UNL and dictionary attributes of UWs in a UNL relation having an

adjective. The implementation process of adjective morphology is illustrated with the help of an example UNL relation (5.20).

mod(flower.@pl, beautiful(mod<thing)) …(5.20)

Here, UW '*beautiful(mod<thing)'* is identified as an adjective with lexical attribute *'AdjA'* and HW ਸੋਹਣ *sōhaṇ 'beautiful'* from Punjabi-UW dictionary. Its gender and number information is obtained from the head noun that is participating with it in *'mod'* relation. The UW *'flower'* is identified as head noun and it has *'MALE'* and *'@pl'* attributes in Punjabi-UW dictionary and UNL expression, respectively.

The rule given in (5.21) will be fired to generate the adjective morphology for the example adjective ਸੋਹਣ *sōhaṇ 'beautiful'*.

ੇ, @pl,@MALE,@AdjA …(5.21)

With the firing of this rule, ਸੋਹਣ *sōhaṇ 'beautiful'* is changed to ਸੋਹਣੇ *sōhṇē 'beautiful'* in the generation process. In case of UNL expression given in (5.22), the rule given in (5.23) will be fired to generate ਸੋਹਣੀਆਂ *sōhṇīāṃ 'beautiful'* from headword ਸੋਹਣ *sōhaṇ 'beautiful'*.

mod(picture(icl>art).@pl, beautiful(mod<thing)) …(5.22)

ੀਆਂ, @pl,@FEMALE,@AdjA …(5.23)

The example for an adjective that does not change its form is given in (5.24).

aoj(clean(aoj>thing), water(icl>liquid)) …(5.24)

Here, *'clean(aoj>thing)'* is identified as adjective with headword ਸਾਫ਼ *sāf 'clean'*, without attribute *'AdjA'*, from Punjabi-UW dictionary. It signifies that headword ਸਾਫ਼ *sāf 'clean'* will retain its form in the generation process.

The generation rule base for adjective morphology is given in Table 5.4.

Table 5.4: Generation of adjective morphology rules

S.No	Adjective Morphology Rules
1.	ਾ,@MALE,@AdjA
2.	ੇ,@pl,@MALE,@AdjA
3.	ੇ,@pl,@respect,@AdjA
4.	ੀ,@FEMALE,@AdjA

5.	ੀਆਂ,@pl,@FEMALE,@AdjA

An algorithm has again been developed to implement the adjective morphology. This algorithm 5.2 is presented below.

Algorithm 5.2: Processing of adjective morphology rule base

(i) Obtain gender and number information from head noun acting as UW in a UNL relation with the adjective and store this information into an attribute list. If number information is not present, consider the number as singular.

(ii) Search the adjective morphology rule base and fire the rule that has maximum similarity with the elements of attribute list.

5.7.3.3 Pronoun morphology

Pronouns are inflected by case, tense, number and gender information in a sentence. It has been observed that pronoun morphology also depends on UNL relation labels. The pronoun morphology for Punjabi language is presented in this section.

Inflection of pronouns on the basis of case and number

Pronouns are inflected by case and number, *e.g.*, personal pronoun ਮੈਂ *maiṃ 'i'*, changes its form to ਮੈਨੂੰ *mainūṃ*, ਮੈਥੋਂ *maithōṃ*, 'ਮੇਰੇ ਲਈ' *'mērē laī'*, 'ਮੇਰੇ ਕੋਲੋਂ' *'mērē kōlōṃ'*, 'ਮੇਰੇ ਲਾਗਿਓਂ' '*mērē lāgiōṃ' etc.* for singular number and to ਅਸੀਂ *asīṃ*, ਸਾਨੂੰ *sānūṃ*, ਸਾਥੋਂ *sāthōṃ*, 'ਸਾਡੇ ਕੋਲੋਂ' *'sāḍē kōlōṃ'*, 'ਸਾਡੇ ਲਾਗਿਓਂ' '*sāḍē lāgiōṃ' etc.* for plural number depending on the case information of the sentence (Gill, 2008).

Inflection of pronouns on the basis of UNL relation label

In a UNL based generation system, inflections of pronouns are also generated on the basis of UNL relation labels used in a UNL expression (Hrushikesh, 2002). The inflections of pronouns based on UNL relation labels are discussed in this section.

***'agt'* Relation**

Morphology of the relation *agt*(UW1, UW2), where UW2 is a pronoun, depends upon the tense information provided by UNL attributes of UW1. Let us illustrate this concept, with example sentences given below.

Example English sentence with past tense: He ate mangoes. …(5.25)

UNL expression for this example sentence is given in (5.26).

{unl}

agt(eat.@past.@entry, he(icl>person))

obj(eat.@past.@entry, mango.@pl)

{/unl} …(5.26)

Equivalent Punjabi sentence: ਉਸ ਨੇ ਅੰਬ ਖਾਧੇ । …(5.27)

Transliterated Punjabi sentence: *us nē amb khādhē.*

Example English sentence with present tense: He eats mangoes. ...(5.28)

UNL expression for this example sentence is given in (5.29).

{unl}

agt(eat.@present.@entry, he(icl>person))

obj(eat.@present.@entry, mango.@pl)

{/unl} …(5.29)

Equivalent Punjabi sentence: ਉਹ ਅੰਬ ਖਾਂਦਾ ਹੈ । …(5.30)

Transliterated Punjabi sentence: *uh amb khāndā hai.*

Example English sentence with future tense: He will eat mangoes. …(5.31)

UNL expression for this example sentence is given in (5.32).

{unl}

agt(eat.@future.@entry, he(icl>person))

obj(eat.@future.@entry, mango.@pl)

{/unl} …(5.32)

Equivalent Punjabi sentence: ਉਹ ਅੰਬ ਖਾਏਗਾ । …(5.33)

Transliterated Punjabi sentence: *uh amb khāēgā.*

From the Punjabi outputs given in (5.27), (5.30) and (5.33), it is evident that in case of *'agt'* relation, pronoun ਉਹ *uh* for the UW *'he(icl>person)'* retains its original form for present and future tense sentences, while for past tense it changes to 'ਉਸ ਨੇ' *'us nē'*.

'*ben'* Relation

In case of *'ben'* relation of type *ben*(UW1, UW2), where UW2 is a pronoun, the morphology of pronoun depends on its number information. Let us illustrate this concept with example sentences given below.

Example English sentence with singular number: He gave a book to him. …(5.34)

UNL expression for this example sentence is,

{unl}

agt(give.@past.@entry, he(icl>person))

obj(give.@past.@entry, book)

ben(give.@past.@entry, he(icl>person))

{/unl} …(5.35)

Equivalent Punjabi sentence: ਉਸ ਨੇ ਉਸ ਨੂੰ ਕਿਤਾਬ ਦਿੱਤੀ । …(5.36)

Transliterated Punjabi sentence: *us nē us nūṃ kitāb dittī.*

Example English sentence for plural number: He gave a book to them. ...(5.37)

UNL expression for this example sentence is,

{unl}

agt(give.@past.@entry, he(icl>person).@pl))

obj(give.@past.@entry, book)

ben(give.@past.@entry, he(icl>person).@pl)

{/unl} ...(5.38)

Equivalent Punjabi sentence: ਉਸ ਨੇ ਉਹਨਾਂ ਨੂੰ ਕਿਤਾਬ ਦਿੱਤੀ । …(5.39)

Transliterated Punjabi sentence: *us nē uhnāṃ nūṃ kitāb dittī.*

From the Punjabi outputs given in (5.36) and (5.39), it is evident that in case of *'ben'* relation, pronoun ਉਹ *uh 'he'* for the UW *'he(icl>person)'* changes to 'ਉਸ ਨੂੰ *'us nūṃ'* and 'ਉਹਨਾਂ ਨੂੰ *'uhnāṃ nūṃ'* depending upon its number information.

***'mod', 'pos'* and *'pof'* Relations**

In case of *'mod', 'pos'* or *'pof'* relations of type *rel*(UW1, UW2), where *'rel'* is '*mod', 'pos'* or '*pof*' and UW2 is a pronoun, the pronoun morphology depends on gender and number information of UW1. Let us illustrate this concept, with example sentences.

Example English sentence: I ate his fruits. ...(5.40)

UNL expression for this example sentence is,

{unl}

agt(eat.@past.@entry, I(icl>person))

obj(eat.@past.@entry, fruit.@pl)

pos(fruit.@pl, he(icl>person))

{/unl} ...(5.41)

Equivalent Punjabi sentence: ਮੈਂ ਉਸ ਦੇ ਫਲ ਖਾਧੇ । …(5.42)

Transliterated Punjabi sentence: *maiṃ us dē fal khādhē.*

Example English sentence: I ate his chapattis. …(5.43)

UNL expression for this example sentence is,

{unl}

agt(eat.@past.@entry, I(icl>person))

obj(eat.@past.@entry, chapatti.@pl)

pos(chapatti.@pl, he(icl>person))

{/unl} …(5.44)

Equivalent Punjabi sentence: ਮੈਂ ਉਸ ਦੀਆਂ ਰੋਟੀਆਂ ਖਾਧੀਆਂ । …(5.45)

Transliterated Punjabi sentence: *maiṃ us dīāṃ rōṭīāṃ khādhīāṃ.*

From the Punjabi outputs given in (5.42) and (5.45), it is evident that in case of *'pos'* relation, pronoun ਓਹ *uh* for the UW *'he(icl>person)'* changes to 'ਉਸ ਦੇ' '*us dē'* for UW1 *'fruit'* having UNL attribute *'@pl'* and lexical attribute *'MALE'* (from Punjabi-UW dictionary) in UNL expression given in (5.42) and pronoun ਓਹ *uh* for UW *'he(icl>person)'* changes to 'ਉਸ ਦੀਆਂ' '*us dīāṃ*' for UW1 *'chapatti'* having UNL attribute *'@pl'* and lexical attribute *'FEMALE'* (from Punjabi-UW dictionary) in UNL expression given in (5.44).

Implementation of pronoun morphology

As discussed earlier the pronoun morphology of a node depends upon case, tense, number and gender information of both UWs and on the UNL relation participated by the node in a UNL expression. Thus, accordingly a four column rule base has been prepared for pronoun morphology. The rule format for pronoun morphology is given in (5.46).

UNL relation:HW:Attribute1+Attribute2+… :morphed pronoun …(5.46)

Here, each column is separated by ':'; first column of the rule base is the UNL relation that represents relation participated by the pronoun corresponding to which the reference to rule base is being made; second column represent the headword of the pronoun; third column contains the list of attributes whose presence needs to be asserted for firing of rule (if there are more than one attribute that needs to be asserted, then they are separated

by '+' sign), *i.e.*, the rule will be fired if attributes present in this column matches with attributes of UWs in the UNL expression and their dictionary attributes; fourth column contains the morphed pronoun that needs to be generated for the conditions specified in the rule. The implementation process of pronoun morphology is illustrated with the help of an example UNL relation (5.47).

ben(give.@past.@entry, he(icl>person).@pl) …(5.47)

Here, UW *'he(icl>person)'* is identified as a pronoun having HW ਉਹ *uh 'he'* and *'@pl'* as a UNL attribute in the UNL relation *'ben'*. The rule given in (5.48) will be fired to generate the pronoun morphology for the HW ਉਹ *uh 'he'*.

ben:ਉਹ:@pl:ਉਹਨਾਂ ਨੂੰ …(5.48)

It means that for *'ben'* relation the pronoun ਉਹ *uh 'he'* changes to ਉਹਨਾਂ ਨੂੰ *'uhnāṃ nūṃ'* for *'@pl'* attribute.

The generation rule base for pronoun morphology is given in Table 5.5.

Table 5.5: Generation of pronoun morphology rules

S.No	Pronoun morphology rules
1.	agt:ਉਹ:@lnk:ਉਹ
2.	agt:ਉਹ:@present+@ability:ਉਹ
3.	agt:ਉਹ:@past+@custom:ਉਹ
4.	agt:ਉਹ:@present+@custom:ਉਹ
5.	agt:ਉਹ:@future+@custom:ਉਹ
6.	agt:ਉਹ:@past+@ability:ਉਹ
7.	agt:ਉਹ:@future+@ability:ਉਹ
8.	agt:ਉਹ:@past+@complete:ਉਹ
9.	agt:ਉਹ:@present+@complete:ਉਹ
10.	agt:ਉਹ:@future+@complete:ਉਹ
11.	agt:ਉਹ:@past+link:ਉਸ ਨੇ
12.	agt:ਉਹ:@future+link:ਉਸ ਨੇ
13.	agt:ਉਹ:@present+link:ਉਸ ਨੇ

14.	agt:ਉਹ:@present+iya:ਉਸ ਨੇ
15.	agt:ਉਹ:@present:ਉਹ
16.	agt:ਉਹ:@progress:ਉਹ
17.	agt:ਉਹ:@grant:ਉਹ
18.	agt:ਉਹ:@past+@repeat:ਉਹ
19.	agt:ਉਹ:@past+@vaa:ਉਹ
20.	agt:ਉਹ:@past+@ja:ਉਹ
21.	agt:ਉਹ:@past:ਉਸ ਨੇ
22.	agt:ਉਹ:@future:ਉਹ
23.	agt:ਉਹ:@complete:ਉਸ ਨੇ
24.	agt:ਉਹ:@present:ਉਹ
28.	ben:ਤੂੰ:..@future:ਤੇਰੇ ਲਈ
29.	ben:ਤੂੰ:..@:ਤੇਰੇ ਲਈ
30.	agt:ਤੂੰ:..@pl:ਤੁਸੀਂ
31.	obj:ਤੂੰ:..@pl:ਤੁਹਾਨੂੰ
32.	obj:ਤੂੰ:@:ਤੁਹਾਨੂੰ
33.	obj:ਮੈਂ:..@pl:ਸਾਨੂੰ
34.	obj:ਮੈਂ:..@:ਮੈਨੂੰ
35.	pos:ਮੈਂ:@FEMALE+@pl:ਮੇਰੀਆਂ
36.	pos:ਮੈਂ:@FEMALE:ਮੇਰੀ
37.	pos:ਮੈਂ:@MALE+@pl:ਮੇਰੇ
38.	pos:ਮੈਂ:@MALE:ਮੇਰਾ
39.	bas:ਤੂੰ:null:ਤੇਰੇ
40.	agt:ਮੈਂ:@pl:ਅਸੀਂ
41.	obj:ਉਹ:@sg:ਉਸ ਨੂੰ
42.	pos: ਉਹ:@MALE+@sg:ਉਸ ਦਾ

43.	pos: ਉਹ:@MALE+@pl:ਉਸ ਦੇ
44.	pos: ਉਹ:@FEMALE+@sg:ਉਸ ਦੀ
45.	pos: ਉਹ:@FEMALE+@pl:ਉਸ ਦੀਆਂ
46.	ben: ਉਹ:@sg:ਉਸ ਨੂੰ
47.	ben: ਉਹ:@pl: ਉਹਨਾਂ ਨੂੰ

5.7.3.4 Verb morphology

Punjabi verbs are classified as main verbs and auxiliary verbs. The main verbs have transitivity and causativity assigned to them. Transitive verbs are those that require an object in a sentence while intransitive verbs do not require an object. For example, ਪੜ੍ਹ *paṛh 'study'* is transitive as in 'ਉਸ ਨੇ ਕਿਤਾਬ ਪੜ੍ਹੀ' *'usnē kitāb paṛhī' 'He read the book'* and verb ਦੌੜ *dauṛ 'run'* is intransitive as in 'ਮੁੰਡਾ ਦੌੜ ਰਿਹਾ ਸੀ' *'muṇḍā dauṛ rihā sī' 'The boy was running'*. There are two types of causatives used in Punjabi. These are simple causative and double causative. In general, a simple causative is formed by adding ਆ *ā* and double causative is formed by adding ਵਾ *vā*. For example, causative forms of verb root ਪੜ੍ਹ *paṛh 'study'* are ਪੜਾ *paṛā* (simple causal) and ਪੜਵਾ *paṛvā* (double causal). All transitive, intransitive and causative verbs are also known as simple verbs. In Punjabi, there are two auxiliary verbs ਹੈ *hai* for present tense and ਸੀ *sī* for past tense. For future tense in sentences, *'EGA'* form of main verb is used.

Conjunct verbs

Expressing the concept of a given word, in source language, may require two or more words in target language. Many verbs in English can be translated into Punjabi, only by using a noun-verb sequence (*e.g.*, ਸ਼ੁਰੂ ਕਰਨਾ *shurū karnā 'start'*, ਦਿਖਾਈ ਦੇਣਾ *dikhāī dēṇā 'visible'*) or only by using an adjective-verb sequence (*e.g.*, ਮਿੱਠਾ ਲੱਗਣਾ *miṭṭhā laggṇā 'sweet'*, ਚੰਗਾ ਲੱਗਣਾ *caṅgā laggṇā 'like'*) or only by using an adverb-verb sequence (*e.g.*, ਦੂਰ ਹਟਾਉਣਾ *dūr haṭāuṇā 'remove'*) (Chakrabarti *et al.*, 2006). Morphological attributes of these verbs remain same as other verbs that do not contain such a sequence. All inflections are marked only on the verb, and noun or adjective in the sequence remains

uninflected. It has been noted that, many of these verbs are formed by adding the verbs- ਕਰ *kar 'do'* (*e.g.*, ਪੜ੍ਹਾਈ ਕਰ *paṛ̱hāī kar 'study'*) or ਹੋ *hō 'be'* (*e.g.* ਖ਼ਤਮ ਹੋ *k̲h̲atam hō 'finish' etc.*) to the nouns or adjectives. Conjunct verbs are represented in the lexicon by attribute *'CV'*, the verbs that are formed by adding the verb ਕਰ *kar 'do'* to nouns or adjectives are represented by *'link'*, and those that are formed by adding the verb ਹੋ *hō 'be'* to nouns or adjectives are represented by *'lnk'* attributes in the lexicon (Singh *et al.*, 2007) as shown in Punjabi-UW dictionary entries given in (5.49).

[ਨਿਯੁਕਤ] {}"appoint(agt>thing,obj>person)" (V,Va,CV,link) <P,0,0>;/(*niyukat*)

[ਪ੍ਰਾਪਤ] {} "achieve(icl>attain)" (V,CV,link,Va) <P,0,0>; /(*prāpat*)

[ਹਾਜ਼ਰ] {}"attend(agt>thing,obj>place)" (V,lnk,Va) <P,0,0>;/(*hāzar*)

[ਕਾਰਨ] {}"cause(aoj>thing,obj>thing)" (V,lnk,Va) <P,0,0>;/(*kāran*) …(5.49)

Verb paradigms

Verbs play a vital role in extracting the Gender, Number, Person, Tense, Aspect and Modality (GNPTAM) information from a given sentence. In order to capture the morphological variations of verbs, they are categorized into various paradigms to identify uninflected forms of words that share similar inflections (Singh *et al.*, 2007). There are approximately one hundred verb paradigms classified based on their vowel ending and GNPTAM information (Gill and Gleason, 1986; Singh and Singh, 1980). Some important verb paradigms for first person singular masculine gender are given in Appendix-C.

Verb morphology for UNL attributes

The verb morphology is generated on the basis of UNL attributes during the DeConversion process. The rules for verb morphology can, generally, be classified into three types as given below.

(a) General verb morphology rules,

(b) Verb morphology rules for passive sentence and

(c) Verb morphology rules for conjunct verbs.

The information about GNPTAM is generally extracted from subject of a sentence. For passive sentences this information is extracted from the object of the sentence. In this situation, the UNL attribute *'@passive'* is associated with the main verb and *'@topic'* is associated with the object of the sentence (Nalawade, 2007). The verb morphology for

conjunct verbs is given in Table 5.6 for ਕਰ *kar 'do'* type of verbs and Table 5.7 for ਹੋ *hō 'be'* type of verbs.

Table 5.6: Some examples of conjunct verb morphology for ਕਰ *kar 'do'* type of verbs

UNL Attributes	Verb morphology
@custom,@present,@sg,@male,@link	ਕਰਦਾ ਹੈ *karadā hai*
@past,@sg,@male,@link	ਕੀਤਾ *kītā*
@future,@sg,@male,@link	ਕਰੇਗਾ *karēgā*
@custom,@past,@sg,@male,@link	ਕਰਦਾ ਸੀ *karadā sī*
@progress,@present,@sg,@male,@link	ਕਰ ਰਿਹਾ ਹੈ *kar rihā hai*
@progress,@past,@sg, @male,@link	ਕਰ ਰਿਹਾ ਸੀ *kar rihā sī*
@progress,@future,@sg, @male,@link	ਕਰ ਰਿਹਾ ਹੋਵੇਗਾ *kar rihā hōvēgā*
@repeat,@past,@sg,@male,@link	ਕਰਦਾ ਰਹਿੰਦਾ ਸੀ *karadā rahindā sī*
@repeat,@future,@sg,@male,@link	ਕਰਦਾ ਰਹੇਗਾ *karadā rahēgā*
@present,@complete,@sg,@male,@link	ਕੀਤਾ ਹੈ *kītā hai*
@future,@complete,@sg,@male,@link	ਕਰ ਚੁੱਕਿਆ ਹੋਵੇਗਾ *kar cukkiā hōvēgā*
@obligation, @present,@sg,@male,@link	ਕਰਨਾ ਪੈਂਦਾ ਹੈ *karnā paindā hai*

@obligation,@past, @sg,@male,@link	ਕਰਨਾ ਪਿਆ *karnā piā*
@obligation,@future,@sg,@male,@link	ਕਰਨਾ ਪਵੇਗਾ *karnā pavēgā*
@ability,@present,@sg,@male,@link	ਕਰ ਸਕਦਾ ਹੈ *kar sakdā hai*
@ability,@past,@sg,@male,@link	ਕਰ ਸਕਦਾ ਸੀ *kar sakdā sī*
@ability,@future,@sg,@male,@Va,@link	ਕਰ ਸਕਦਾ ਹੋਵੇਗਾ *kar sakdā hōvēgā*

Table 5.7: Some examples of conjunct verb morphology for ਹੋ *hō 'be'* type of verbs

UNL attributes	Verb morphology
@custom,@present,@sg,@male,@lnk	ਹੁੰਦਾ ਹੈ *hundā hai*
@past,@sg,@male,@lnk	ਹੋਇਆ *Hōiā*
@future,@sg,@male,@lnk	ਹੋਵੇਗਾ *Hōvēgā*
@custom,@past,@sg,@male,@lnk	ਹੁੰਦਾ ਸੀ *hundā sī*
@progress,@present,@sg,@male,@lnk	ਹੋ ਰਿਹਾ ਹੈ *hō rihā hai*
@progress,@past,@sg,@male,@lnk	ਹੋ ਰਿਹਾ ਸੀ *hō rihā sī*
@progress,@future,@sg,@male,@lnk	ਹੋ ਰਿਹਾ ਹੋਵੇਗਾ *hō rihā hōvēgā*
@repeat,@present,@sg,@male,@lnk	ਹੁੰਦਾ ਰਹਿੰਦਾ ਹੈ

	hundā rahindā hai
@repeat,@future,@sg,@male,@lnk	ਹੁੰਦਾ ਰਹੇਗਾ *hundā rahēgā*
@present,@complete,@sg,@male,@lnk	ਹੋ ਚੁੱਕਿਆ ਹੈ *hō cukkiā hai*
@past,@complete,@sg,@male,@lnk	ਹੋ ਚੁੱਕਿਆ ਸੀ *hō cukkiā sī*
@future,@complete,@sg,@male,@lnk,	ਹੋ ਚੁੱਕਿਆ ਹੋਵੇਗਾ *hō cukkiā hōvēgā*
@present,@ability,@sg,@male,@lnk	ਹੋ ਸਕਦਾ ਹੈ *hō sakdā hai*
@past,@ability,@sg,@male,@lnk	ਹੋ ਸਕਦਾ ਸੀ *hō sakdā sī*
@future,@ability,@sg,@male,@lnk	ਹੋ ਸਕੇਗਾ *hō sakēgā*

When a sentence does not have a main verb in it, then auxiliary verb acts like the main verb. These sentences have a predicative adjective in it and require a verb terminator at the end. They have *'aoj'* relation between *UW1* and *UW2*; and *'@pred'* attribute is used with the *UW1* for its UNL expression (Dwivedi, 2002).

In the next section function word insertion phase of Punjabi DeConverter has been discussed.

5.8 Function word insertion phase

Function word insertion phase is used to insert function words like case markers or postpositions and conjunctions in Punjabi (*e.g.*, ਨੇ *nē*, ਨੂੰ *nūṃ*, ਉੱਤੇ *uttē 'over'*, ਦਾ *dā 'of'*, ਕੋਲੋਂ *kōlōṃ 'from'*, ਅਤੇ *'and' etc.*) to the morphed words generated by morphology phase. Insertion of function words in the generated output depends upon UNL relation and the conditions imposed on parent and child nodes' attributes in a relation (Singh *et al.*, 2007). A rule base has been prepared for this purpose. For each of the forty six UNL relations,

different function words are used depending upon the grammatical details of the target language (Dey and Bhattacharyya, 2005; Birendra and Sanat, 2005). An exhaustive study on *'kaarak'* system of Punjabi language with respect to UNL relations and function words has been carried out in this work (Somal *et al.*, 2005). The findings of this study are presented in Table 5.8.

Table 5.8: '*kaarak'* system with respect to UNL relations and function words

Classical case	*kaaraks*	UNL relations	Punjabi function words
Nominative case	*karta kaarak*	*'agt', 'cag', 'ptn', 'cao'*	ਨੇ *nē*
Accusative case	*karma kaarak*	*'obj', 'ben', 'cob', 'per', 'to'*	ਨੂੰ *nūṃ*
Instrumental case	*karan kaarak*	*'ins', 'met', 'and', 'or', 'coo', 'ptn', 'cao', 'cag', 'man'*	ਨਾਲ਼ *nāḷ*, ਦੁਆਰਾ *duārā*, ਰਾਹੀਂ *rāhīṃ*, ਤੋਂ *tōṃ*
Dative case	*sampradaan kaarak*	*'ben', 'gol', 'pur', 'rsn'*	ਨੂੰ *nūṃ*, ਲਈ *laī*, ਵਾਸਤੇ *vāsatē*, ਹਿਤ *hit*, ਕਾਰਨ *kāran*
Ablative case	*apaadaan kaarak*	*'frm', 'src', 'plf', 'tmf'*	ਵਿਚੋਂ *vicōṃ*, ਤੋਂ *tōṃ*, ਉੱਤੇ *uttē*, ਹੇਠੋਂ *hēṭhōṃ*
Genitive case	*sambandh kaarak*	*'mod', 'pof', 'seq', 'icl', 'con', 'bas', 'obj'*	ਦਾ *dā*, ਦੇ *dē*, ਦੀ *dī*, ਦੀਆਂ *dīāṃ*
Case of time-place	*adhikaran kaarak*	*'via', 'plt', 'tim', 'tmt', 'to', 'gol', 'scn', 'opl', 'dur', 'fmt', 'plc'*	ਵਿਚ *vic*, ਉੱਤੇ *uttē*, ਨੇੜੇ *nēṛē*, ਵੱਲੋਂ *vallōṃ*, ਵੇਲੇ *vēlē*, ਤੱਕ *takk*

There are some UNL relations that are not covered by the *'kaaraks'*. These are: *'cnt'* (content), *'equ'* (equivalent), *'int'* (intersection), *'iof'* (instance-of), *'nam'* (name), *'qua'* (quantity) and *'aoj'* (thing with attribute).

5.8.1 Issues in direct mapping of function words with UNL relations

The function words cannot be mapped directly to UNL relations due to following reasons.

- A given UNL relation may have more than one function words corresponding to it. For example, UNL relation '*pof*' may result into insertion of function word like, ਦਾ/ਦੀ/ਦੇ *dā*/*dī*/*dē* depending upon the lexical properties of parent and child nodes of relation.
- The position of inserting a function word in the generated output is not fixed. The function words are not simply inserted between the parent and child node of a relation. These may be inserted before or after the parent/child node depending upon the conditions.

In order to handle these issues, a rule base has been prepared for insertion of function words in the generated output.

5.8.2 Rule base for function word insertion

Following Sinha (2005a) and Vachhani (2006), a rule base for insertion of function word has been prepared. This rule base consists of nine columns. The description of each column of this rule format is given below.

- First Column: Used to represent relation name
  The name of UNL relation corresponding to which the reference to the rule base is being made is stored in this column.
- Second Column: Used to represent function word preceding parent node
  The function word, that should be inserted before the parent node of the relation in the generated output, is stored in this column.
- Third Column: Used to represent function word following parent node
  The function word, that should be inserted after the parent node of the relation in the generated output, is stored in this column.
- Fourth Column: Used to represent function word preceding child node

The function word, that should be inserted before the child node of the relation in the generated output, is stored in this column.

- Fifth Column: Used to represent function word following child node
  The function word, that should be inserted after the child node of the relation in the generated output, is stored in this column.
- Sixth Column: Used to represent positive conditions for parent node
  The attributes whose presence needs to be asserted on the parent node for firing of the rule is stored in this column.
- Seventh Column: Used to represent negative conditions for parent node
  The attributes whose absence needs to be asserted on the parent node for firing of the rule is stored here.
- Eighth Column: Used to represent positive conditions for child node
  The attributes whose presence needs to be asserted on the child node for firing of the rule is stored in this column.
- Ninth Column: Used to represent negative conditions for child node
  The attributes whose absence needs to be asserted on the child node for firing of the rule is stored here.

If there are more than one attributes that need to be asserted on a given node for firing of a rule, then they are stored in the rule base with the separation of '**#**' sign. Here, the attributes represent the UNL attributes (obtained from given UNL expression) or the lexical attributes (obtained from Punjabi-UW dictionary) of the node.

The rule base for function word insertion is illustrated with an example rule given in (5.50).

agt:null:null:null:ਨੇ:@past#V:VINT#@progress#jA:N#3rd:1st#2nd ...(5.50)

Here, *'agt'* is a UNL relation under consideration, and the firing of given rule will result into the insertion of function word ਨੇ *nē* following the child node in the generated output, because function word appears in the fifth column and second, third and fourth columns contain *'null'* in the rule. The sixth column contains *'@past#V'*, which means that the rule will be fired if the parent of *'agt'* relation contains *'@past'* as its UNL attribute in the given input UNL expression and has a *'V'* as its lexical attribute in the Punjabi-UW dictionary. The seventh column contains *'VINT#@progress#jA'* which refers to the

attributes whose absence need to be asserted on the parent node for firing of the rule. It means that parent node should not contain *'VINT'* (intransitive verb), *'jA'* (*'go'* verb) attributes in the lexicon and '*@progress'* attribute in the parent of UNL expression. The eighth column of the rule given in (5.50) contains *'N#3rd'* which refers to the attribute, whose presence needs to be asserted on the child node for firing of the rule, *i.e.*, child should have a *'N'* (noun) and *'3rd'* (third person) attribute in the Punjabi-UW dictionary. Ninth column contains *'1st#2nd'* which refers to the attribute whose absence needs to be asserted on the child node for firing of the rule. It means that child node should not refer to first person or second person in the sentence. Thus, if relation *'agt'* has a parent node with *'@past'* and *'V'* attribute; and without *'VINT'*, *'jA'*, '*@progress'* and *'@custom'* attributes; and has a child node with *'N'* and *'3rd'* attribute and without *'1st'* and *'2nd'* attribute, then function word ਨੇ *nē* will be inserted following the child node in the generated output.

For example, in '*agt(eat(icl>do).@past.@entry, boy(icl>male child))'*, the parent node of relation '*agt'* is '*eat'* having '*V'* and '*@past'* attribute and do not have '*VINT'* attribute in the lexicon. The child node of '*agt'* relation is '*boy(icl>male child)'* that has '*N'* and *'3rd'* attribute and does not has *'1st'* and *'2nd'* attribute in the lexicon. As such, this will result into the firing of rule (5.50) and will thus result into the generation of function word ਨੇ *nē* followed by child node '*boy(icl>male child)'* in the generated output.

The complete rule-base for function word insertion is given in Appendix-D.

5.8.3 Implementation of the function word insertion rule base

An algorithm has been developed to implement the insertion of function words in the generated output. This algorithm 5.3 is presented below.

Algorithm 5.3: Processing of function word insertion rule base

(i) Process the linked list consisting of all UNL relations that are present in the input UNL expression. This list called as RelationList is created by PseudoCode 5.1.

(ii) Perform steps (iii) to (vii) for each element of RelationList until the complete list is traversed.

(iii) Obtain parent node and child node of UNL relation from the NodeList.

(iv) Obtain UNL attributes and lexical attributes of parent and child nodes of given UNL relation and store this information in parent attribute list and child attribute list, respectively.

(v) For each rule perform the following steps.

(a) If a given UNL relation matches with the first column of the rule then perform steps (b) to (f), otherwise skip the rule.

(b) Compare the attributes of rules given in sixth column with the parent attribute list, if all the attributes given in the sixth column of rule base appear in the parent attribute list, then set parent positive condition match flag to 1 otherwise to 0.

(c) Compare the attributes of rules given in seventh column with the parent attribute list, if all the attributes given in the seventh column of rule base do not appear in the parent attribute list, then set parent negative condition match flag to 1 otherwise to 0.

(d) Compare the attributes of rules given in eighth column with the child attribute list, if all the attributes given in the eighth column of rule base appear in the child attribute list, then set child positive condition match flag to 1 otherwise to 0.

(e) Compare the attributes of rules given in ninth column with the child attribute list, if all the attributes given in the ninth column of rule base do not appear in the child attribute list, then set child negative condition match flag to 1 otherwise to 0.

(f) If all the flags, *i.e.*, parent positive condition match flag, parent negative condition match flag, child positive condition match flag and child negative condition match flag are 1, then the corresponding rule will be marked for firing.

(vi) If no rule is marked for firing then go to step (ii).

(vii) For each of the rule marked for firing performs steps (viii) to (xi) and then go to step (ii).

(viii) If second column of the marked rule is not null, then append the function word given in second column at the position preceding to the Punjabi word attribute of parent node.

(ix) If third column of the marked rule is not null, then append the function word given in third column at the position next to the Punjabi word attribute of the parent node.

(x) If fourth column of the marked rule is not null, then append the function word given in fourth column at the position preceding to the Punjabi word attribute of child node.

(xi) If fifth column of the marked rule is not null, then append the function word given in fifth column at the position following to the Punjabi word attribute of the child node.

Working of this algorithm is now explained with the help of an example sentence given in (5.51).

Example sentence: The computer translated from English to Punjabi. …(5.51)

UNL expression for this example sentence is,

{unl}

agt(translate(icl>do).@past.@entry, computer(icl>machine))

src(translate(icl>do).@past.@entry, english(icl>language))

gol(translate(icl>do).@past.@entry, punjabi(icl>language))

{/unl} …(5.52)

The UNL expression given in (5.52) is processed by morphology phase to produce a node-list given in (5.53).

$Node_1$: Punjabi word: ਅਨੁਵਾਦ ਕੀਤਾ *anuvād kītā*; UW: translate(icl>do)

$Node_2$: Punjabi word: ਕੰਪਿਊਟਰ *kampiūṭar*; UW: computer(icl>machine)

$Node_3$: Punjabi word: ਅੰਗਰੇਜ਼ੀ *aṅgrēzī*; UW: english(icl>language)

$Node_4$: Punjabi word: ਪੰਜਾਬੀ *pañjābī*; UW: punjabi(icl>language) ...(5.53)

In the UNL expression given in (5.52), '*agt'* relation has parent node *'translate'* with *'@past'* attribute obtained from UNL expression and *'V'* attribute obtained from Punjabi-UW dictionary. The child node *'computer(icl>machine)'* has *'N'* (noun) and *'3rd'* (third person) attribute obtained from lexicon and it does not has *'1st'* and *'2nd'* attribute. All the flags, *i.e.*, parent positive condition match flag, parent negative condition match flag,

child positive condition match flag and child negative condition match flag will be set for the rule given in (5.50) and it will append the function word ਨੇ *nē* at the position next to the Punjabi word attribute of child node *'computer(icl>machine)', i.e.*, Punjabi word attribute of child node *'computer(icl>machine)'* will become 'ਕੰਪਿਊਟਰ ਨੇ' ' *kampiūṭar nē'*. For relation '*s*rc', the rule given in (5.54) will be fired since there is only one condition which is to be asserted on the child node and it has *'N'* attribute. Here, child node of *'src'* relation *'english(icl>language)'* has *'N'* attribute in the lexicon. The firing of rule (5.54) will append the function word ਤੋਂ *tōṃ* at the position next to the Punjabi word attribute of child node *'english(icl>language)'*, because function word ਤੋਂ *tōṃ* appears in the fifth column of the rule base. Thus, Punjabi word attribute of child node *'english(icl>language)'* will become 'ਅੰਗਰੇਜ਼ੀ ਤੋਂ' *'aṅgrēzī tōṃ'*.

src:null:null:null:ਤੋਂ:null:null:N:null …(5.54)

For relation '*gol'*, the rule given in (5.55) will be fired since there is only one condition *'N'* and *'INANI'* (inanimate) which is to be asserted on the attributes of child node. Here, child node of *'gol'* relation *'punjabi(icl>language)'* has *'N'* and *'INANI'* attributes in the lexicon. The rule (5.55) will append the function word ਵਿਚ *vic* at the position next to the Punjabi word attribute of child node *'punjabi(icl>language)'*, because case marker ਵਿਚ *vic* appears in the fifth column of the rule base, *i.e.*, Punjabi word attribute of child node *'punjabi(icl>language)'* will become 'ਪੰਜਾਬੀ ਵਿਚ' *'pañjābī vica'*.

gol:null:null:null:ਵਿਚ:null:null:INANI#N:null …(5.55)

After processing with the fired rules given in (5.50), (5.54) and (5.55), function word insertion phase will modify the nodes given in (5.53) and will result into the nodes given in (5.56).

Node$_1$: Punjabi word: ਅਨੁਵਾਦ ਕੀਤਾ *anuvād kītā*; UW: translate(icl>do)

Node$_2$: Punjabi word: ਕੰਪਿਊਟਰ ਨੇ *kampiūṭar nē*; UW: computer(icl>machine)

Node$_3$: Punjabi word: ਅੰਗਰੇਜ਼ੀ ਤੋਂ *aṅgrēzī tōṃ*; UW: english(icl>language)

Node$_4$: Punjabi word: ਪੰਜਾਬੀ ਵਿਚ *pañjābī vica*; UW: punjabi(icl>language) ..(5.56)

The order of processing of these nodes to define the word order in the generated output is defined by the syntax planning phase of the DeConverter. In the next section, the syntax planning phase of DeConverter has been discussed.

5.9 Syntax planning phase

Syntax planning is the process of linearizing the lexemes in the semantic hyper-graph. As such, it is a process to define the word order in the generated sentence. In a language, some word orders are considered more natural than others. The syntax planning deals with the arrangements of words in the generated output so that output matches with the natural language sentence. The system assigns relative positions to various words based on the relations they share with the headword in a sentence. The structural differences between English (Subject-Verb-Object) and Punjabi (Subject-Object-Verb) languages necessitate the syntax planning phase in the development of a Punjabi DeConverter.

5.9.1 Major issues in syntax planning

The parent child relationship and matrix based priority of relations are two important issues in syntax planning phase. In a UNL binary relation *rel*(UW1, UW2), UW1 acts as the parent of the relation whereas, UW2 acts as the child of the relation. For each parent child relationship, the system should state whether the parent should be ordered before or after the child in the generated output (Vora, 2002). For the syntax plan of Punjabi language, in most of UNL relations the parent node appears right to all of its children nodes in the generated output.

To illustrate this concept, let us consider a UNL relation *agt*(UW1, UW2), where UW1 is a verb and UW2 is the subject or agent of that event. Since, Punjabi is a SOV language, so subject always comes to the left of the verb. Same is the case of *'obj'* relation. In case of a UNL expression given in (5.57) both the children, *i.e.*, *'boy(icl>male child)'* and *'rice(icl>food)'* will be placed left to the parent *'eat'*.

{unl}
agt(eat.@present.@entry, boy(icl>male child))
obj(eat.@present.@entry, rice(icl>food))
{/unl} …(5.57)

Since, both children are inserted to the left, the child inserted first in the generated output will be decided by the matrix based priority of relations.

5.9.2 Matrix based priority of relations

The necessity of matrix based priority of relations occurs when the child of two or more UNL relations has a common parent. It is important to decide the relative positions of children (sharing a common parent) with respect to each other in the generated output. In the proposed Punjabi DeConverter, the relative positions of children with respect to each other are decided by a matrix '*M'* following Vora (2002). This matrix '*M'* has 46 rows and 46 columns, representing 46 UNL relations specified in UNL specifications (Uchida, 2005). This matrix $M=[m_{ij}]$, $i=1, 2, \ldots, 46$; $j=1, 2, \ldots, 46$ contains the elements as *'L', 'R'* and *'-'*. Here, *'L'* means towards left, *'R'* means towards right and *'-'* means no action.

If 'm_{ij}' = *'L'*, then the position of the child of the i^{th} relation is left to the child of the j^{th} relation when the two children share a common parent. If 'm_{ij}' = *'R'*, then the position of the child of i^{th} relation is right to the child of the j^{th} relation when the two children share a common parent. If 'm_{ij}' = *'-'*, then no action is to be taken as it is impossible that child of this i^{th} relation shares a common parent with the child of this j^{th} relation (Hrushikesh, 2002; Vachhani, 2006).

This is illustrated with 'R_i' and 'R_j' as two UNL binary relations between three nodes 'N_1', 'N_2' and 'N_3' as $R_i(N_3, N_1)$ and $R_j(N_3, N_2)$. Here, node 'N_1' and node 'N_2' are the children of same parent 'N_3' as shown in Figure 5.8.

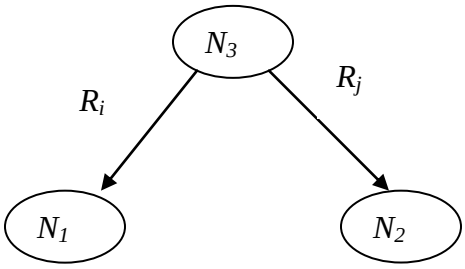

Figure 5.8: UNL graph of two nodes with same parent

Based on the structure of the target language, if 'N_1' appears at the left of 'N_2' in the generated sentence as denoted by '$(N_1\ L\ N_2)$', then the priority matrix shown in Figure 5.9 needs to be maintained for its syntax plan.

	R_i	R_j
R_i	-	L
R_j	R	-

Figure 5.9: Matrix representation for $(N_1\ L\ N_2)$ structure

The priority of the child of the relation depends upon the frequency of '*L'* in its row. If a relation has all '*L'* in its row, then the child node of that relation will have highest priority and it will appear at extreme left in the generated output. Similarly, if a relation has all '*R'* in its row, then the child node of that relation will have lowest priority and will appear at extreme right of all the children sharing the common parent in the generated output (Nalawade, 2007; Vachhani, 2006). Owing to the fact that we can associate a priority to the child associated with a relation, the matrix *'M'* is called as Priority Matrix. This concept is illustrated with an example UNL expression given in (5.52). The UNL graph for this UNL expression is shown in Figure 5.10.

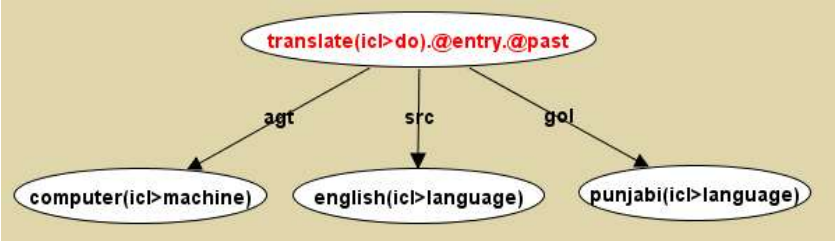

Figure 5.10: UNL graph for UNL expression given in (5.52)

As shown in Figure 5.10, children of '*agt'*, '*src'* and *'gol'* relations have a common parent '*translate(icl>do)'*. According to SOV structure of Punjabi language, the correct word order is produced in the generated output by processing the nodes generated by function word insertion phase in the sequence given in (5.58).

Node$_2$: Punjabi word: ਕੰਪਿਊਟਰ ਨੇ *kampiūṭar nē*; UW: computer(icl>machine)

Node$_3$: Punjabi word: ਅੰਗਰੇਜ਼ੀ ਤੋਂ *aṅgrēzī tōṃ*; UW: english(icl>language)

Node$_4$: Punjabi word: ਪੰਜਾਬੀ ਵਿਚ *pañjābī vica*; UW: punjabi(icl>language)

Node$_1$: Punjabi word: ਅਨੁਵਾਦ ਕੀਤਾ *anuvād kītā*; UW: translate(icl>do) ...(5.58)

It means that, child of *'agt'* relation should appear in the left most position to the child of *'src'* and *'gol'* relations in the generated output and the child of *'src'* relation should appear to the left of child of *'gol'* relation and right of child of *'agt'* relation. In order to produce the correct word order in the generated output, the system should maintain the priority matrix of *'agt'*, *'src'* and *'gol'* relations (because they share a common parent) as shown in Figure 5.11.

	agt	src	gol
agt	-	L	L
src	R	-	L
gol	R	R	-

Priorities of relations
agt: 2
src: 1
gol: 0

Figure 5.11: Priority matrix for syntax plan of UNL expression given in (5.52)

Here, the *'agt'* relation has highest priority, *i.e.*, 2; the *'src'* relation has priority 1 while *'gol'* relation has priority 0. So, the child of *'agt'* relation will be processed first followed by child of *'src'* and *'gol'* relations. After processing all the children nodes, the parent node is processed to generate the correct word order as shown in generated output (5.59) having equivalent English sentence given in (5.60).

ਕੰਪਿਊਟਰ ਨੇ ਅੰਗਰੇਜ਼ੀ ਤੋਂ ਪੰਜਾਬੀ ਵਿਚ ਅਨੁਵਾਦ ਕੀਤਾ । …(5.59)

kampiūṭar nē aṅgrēzī tōṃ pañjābī vic anuvād kītā.

The computer translated from English to Punjabi. …(5.60)

5.9.3 Syntax planning for simple sentences

A simple sentence contains a subject and a verb, and it expresses a complete thought. The UNL expression of simple sentences is converted into a simple node-net or UNL graph by UNL parser. The processing sequence of nodes of this UNL graph controls the word order in the generated output. A program has been developed in Java to control the sequence of processing of nodes of simple UNL graph. The PseudoCode 5.2 contains the instructions corresponding to this program.

PseudoCode 5.2: To control the processing sequence of nodes of simple UNL graph

```
1.  begin
2.      Start traversing the graph from entry node and set this as active node;
3.  while (true)
4.      if (active node has no parent)
5.          if (active node has no unprocessed child)
6.              add node to final string;
7.              Mark node as processed;
8.              exit from the loop;
9.          else
```

```
10.                     if (node is not already marked as visited)
11.                         mark node as visited node;
12.                     end-if
13.                     get highest priority unprocessed child relation;
14.                     set highest priority unprocessed child relation as active node;
15.             end-if
16.         else
17.             if (node has one parent)
18.                     if (active node has no unvisited child)
19.                         add node to final string;
20.                         mark node as processed;
21.                         set parent of node as active node;
22.                     else
23.                         if (node is not already marked as visited)
24.                             mark node as visited node;
25.                         end-if
26.                         get highest priority unprocessed relation child;
27.                         set highest priority unprocessed child relation as active
                            node;
28.                     end-if
29.             end-if
30.         end-if
31.     end-while
32.     end
```

PseudoCode 5.2 performs the processing of nodes of UNL graph according to priority matrix. The given PseudoCode works in a loop, and control exits out of it after the processing of all nodes of the graph. It first traverses the highest priority child node sharing a common parent. If a node has no further unvisited child, then it is processed and added into the final string. After processing of the child node, the parent of that node

becomes active node and again traverses the highest priority un-processed child node. This process continues until the entry node gets processed.

The working of PseudoCode 5.2 is illustrated with an example English sentence given in (5.61).

He has scored 80% marks. …(5.61)

The UNL expression for this example sentence is,

{unl}

agt(score.@present.@complete.@entry, he)

obj(score.@present.@complete.@entry, marks)

mod(marks, percent)

qua(percent, 80)

{/unl} ...(5.62)

UNL graph for UNL expression (5.62) is depicted in Figure 5.12.

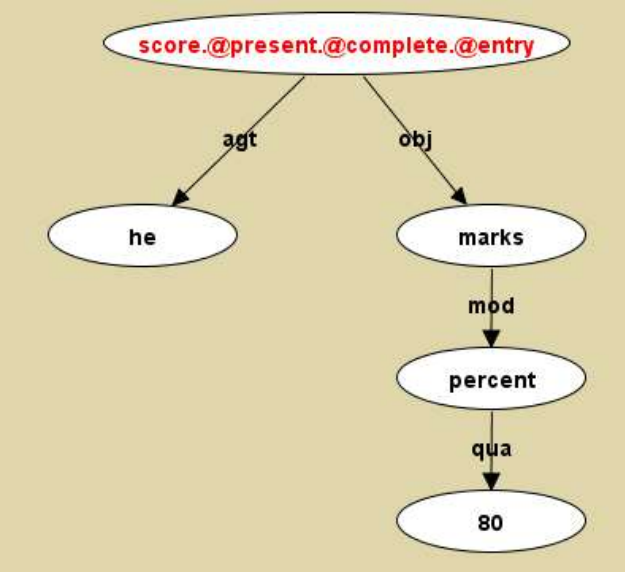

Figure 5.12: UNL graph for UNL expression (5.62)

The node list for the UNL expression given in (5.62) after processing of morphology phase and function word insertion phase is given in (5.63).

Node$_1$: Punjabi word: ਹਾਸਲ ਕੀਤੇ *hāsal kītē*; UW: score

Node$_2$: Punjabi word: ਉਸ ਨੇ *us nē*; UW: he

Node$_3$: Punjabi word: ਅੰਕ *aṅk*; UW: marks

Node$_4$: Punjabi word: ਫ਼ੀਸਦੀ *fīsadī*; UW: percent

$Node_5$: Punjabi word: 80; UW: 80 …(5.63)

In Figure 5.12, the entry node is *'score.@present.@complete.@entry'* as this contains *'@entry'* attribute and the labels on edges indicate UNL relation labels. Here, the priority of the *'agt'* is higher than the priority of the *'obj'* relation as shown in Figure 5.13. Thus *'he'* node will be traversed first. This is worth mentioning here that the priority matrix given in Figure 5.13 is a sub-matrix of the earlier defined matrix *'M'*.

	agt	obj
agt	-	L
obj	R	-

Priorities of relations
agt: 1
obj: 0

Figure 5.13: Priority matrix for *'agt'* and *'obj'* relations

The *'he'* node has no child and its parent node *'score'* is already visited, so it will be processed and its Punjabi word attribute will be appended to the final string used to store generated output, *i.e.*, the final string will become 'ਉਸ ਨੇ'. Now, parent of *'he'* node, *i.e.*, *'score'* node will become the active node. It has only one unprocessed child, *i.e.*, *'marks'* node. So, it will be traversed next and marked as visited. It has one unprocessed child node *'percent'*, so it will be traversed and marked as visited. The *'percent'* node also has one unprocessed child node *'80'*, so it will be traversed next and marked as visited. The node *'80'* has no further unprocessed child, so it will finally be processed and its Punjabi word attribute will be appended to the final string, *i.e.*, the final string will become 'ਉਸ ਨੇ 80'. The parent of *'80'* node, *i.e.*, *'percent'* node now will become active node. The *'percent'* node has no unprocessed child, so it will be processed and its Punjabi word attribute will be appended to the final string, *i.e.*, the final string will become 'ਉਸ ਨੇ 80 ਫ਼ੀਸਦੀ'. Now, the parent of *'percent'* node, *i.e.*, *'marks'* node, will become an active node. The *'marks'* node has no unprocessed child, so it will be processed and its Punjabi word attribute will be appended to the final string, *i.e.*, the final string will become 'ਉਸ ਨੇ 80 ਫ਼ੀਸਦੀ ਅੰਕ'. Now, the parent of *'marks'* node, *i.e.*, *'score'* node, will become an active node. It is an entry node and has no unprocessed child, so it will be processed and its Punjabi word will be appended to the final string, *i.e.*, the final string will become 'ਉਸ ਨੇ 80 ਫ਼ੀਸਦੀ ਅੰਕ ਹਾਸਲ ਕੀਤੇ'. Since, the entry node is processed, the control will exit from the loop and the final output according to the syntax plan will be available in the final string.

Thus, PseudoCode 5.2 will result the processing of node-net in the order given in (5.64) and it will result into Punjabi sentence shown in (5.65) as the generated output of Punjabi DeConverter.

$Node_2\ Node_5\ Node_4\ Node_3\ Node_1$ …(5.64)

ਉਸ ਨੇ 80 ਫ਼ੀਸਦੀ ਅੰਕ ਹਾਸਲ ਕੀਤੇ । …(5.65)

us nē 80 fīsadī aṅk hāsal kītē.

5.9.4 Syntax planning of UNL graph with a scope node

Scope is used to represent a compound universal word or a compound concept. A compound concept is a set of binary relations that are grouped together to express a complex concept. This is defined by adding a compound universal word identifier (compound UW-ID) immediately after the relation label. The compound UW is referred with the ID instead of a universal word.

The syntax planning for sentences with compound concept is a bit different from simple sentences. In this case, UNL graph contains a scope node which itself is a UNL graph as depicted in Figure 5.14.

Algorithm 5.4 has been used to produce the syntax plan for a UNL graph with a scope node.

Algorithm 5.4: Processing the nodes of UNL graph with a scope node

(i) Develop the syntax plan by considering scope node as a single node with PseudoCode 5.2.

(ii) Develop the syntax plan of scope node's UNL graph using PseudoCode 5.2.

(iii) Replace the scope node in the output generated in step (i) with the output generated in step (ii) in order to get final generated sentence.

Syntax planning of UNL graph with a scope node is illustrated by an example sentence given in (5.66).

I went to Delhi 20 days back. …(5.66)

UNL expression of this example sentence is,

{unl}

agt(go.@past.@entry, I(icl>person))

plc(go.@past.@entry, Delhi)

tim(go.@past.@entry, :01)

man:01(back.@entry, day)

qua:01(day, 20)

{/unl} …(5.67)

UNL graph of this UNL expression is depicted in Figure 5.14.

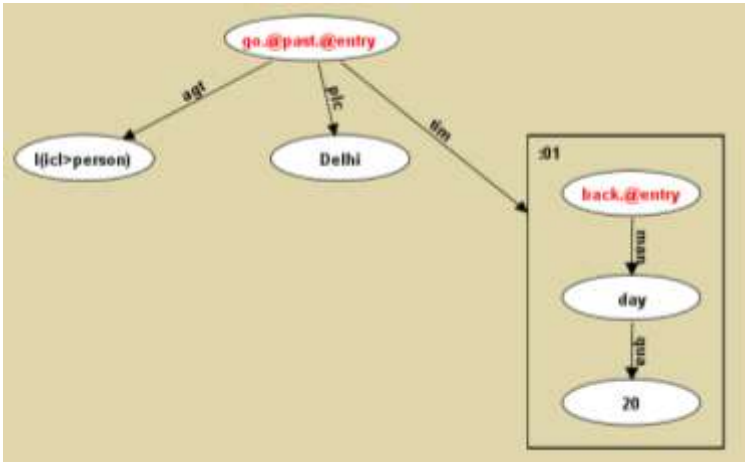

Figure 5.14: UNL graph with a scope node for UNL expression (5.67)

Figure 5.14, *'go'* is entry node having three children, *'I(icl>person)'* with *'agt'* relation; *'Delhi'* with *'plc'* relation and scope node ':01' with '*tim'* relation. Here, *'agt'* relation has highest priority followed by *'tim'* relation and then by *'plc'* relation, as shown in priority matrix in Figure 5.15. Thus, the system will first traverse the *'I(icl>person)'* node, followed by scope node ':01' and then by *'Delhi'* node.

	agt	plc	tim
agt	-	L	L
plc	R	-	R
tim	R	L	-

Priorities of relations
agt: 2
plc: 0
tim: 1

Figure 5.15: Priority matrix for *'agt', 'plc'* and *'tim'* relations

Using PseudoCode 5.2 while considering scope node as a single node, syntax plan of this UNL graph by considering UWs (without constraints) shall be,

I :01 Delhi go …(5.68)

Using algorithm 5.4, the scope node ':01' shown in output (5.68) is replaced by the output generated from the syntax plan of scope node's UNL graph. Again, using PseudoCode 5.2, the syntax plan of UWs (without constraints) of scope node's UNL graph is,

20 day back …(5.69)

Final output for the UNL expression given in (5.67) is generated by replacing scope node in (5.68) with the syntax plan of UWs given in (5.69). Thus, the final syntax plan of UWs (without constraints) for the UNL graph depicted in Figure 5.14 will be generated as given in (5.70). The generated Punjabi sentence after the application of morphology, function word insertion and syntax planning phases of the Punjabi DeConverter is given in (5.71).

I 20 day back Delhi go …(5.70)

ਮੈਂ 20 ਦਿਨ ਪਹਿਲਾਂ ਦਿੱਲੀ ਗਿਆ । …(5.71)

maiṃ 20 din pahilāṃ dillī giā.

5.9.5 Untraversed parent handling

While traversing a UNL graph the system may encounter a situation where, the parent of the node is untraversed. This is illustrated with the help of an example sentence given in (5.72).

Above mentioned DeConverter detail. ...(5.72)

UNL expression for this example sentence is given in (5.73) and corresponding UNL graph is given in Figure 5.16.

{unl}
obj(mention, detail.@entry)
plc(mention, above)
mod(detail.@entry, DeConveter)
{/unl} …(5.73)

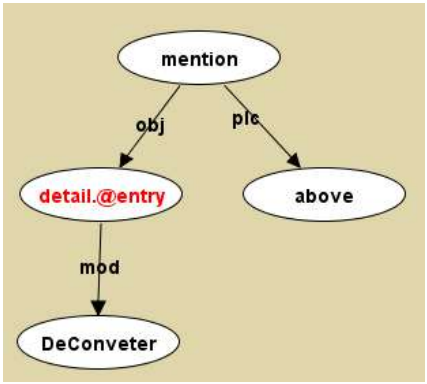

Figure 5.16: UNL graph of UNL expression (5.73)

In UNL graph of Figure 5.16 *'detail'* node acts as the entry node. According to PseudoCode 5.2 the traversal of UNL graph starts from entry node, *i.e.*, *'detail'* node. Here, system encounters a case of untraversed parent, because the parent of *'entry'* node, *i.e.*, *'mention'* node has not been traversed by the system. Following strategy has been implemented for handling an untraversed parent node.

If system encounters a node having untraversed parent, then that node will be removed as child of its parent and it will be added as virtual parent to its untraversed parent. The parent of that node will be set to *'null'* and the untraversed parent node will be set as an active node. Subsequent syntax planning will be carried out according to PseudoCode 5.2. PseudoCode 5.3 has been used to implement this strategy.

PseudoCode 5.3: Handling of untraversed parent node

if (parent of active node is untraversed)

    add active node as virtual parent to its untraversed parent;

    remove active node as child of its parent;

    set active node parent to null;

    set untraversed parent node as active node;

end-if

This PseudoCode 5.3 is inserted between lines numbered 17 and 18 of PseudoCode 5.2 to extend it to handle UNL graph having untraversed parent node.

According to PseudoCode 5.3, for processing of UNL graph depicted in Figure 5.16, the system removes *'detail'* node as the child of its untraversed parent node *'mention'* and sets its parent to null. The system adds *'detail'* node as a virtual parent of its untraversed parent node *'mention'*. The system sets untraversed parent node, *i.e.*, *'mention'* node as an active node. The modified UNL graph after these actions is depicted in Figure 5.17.

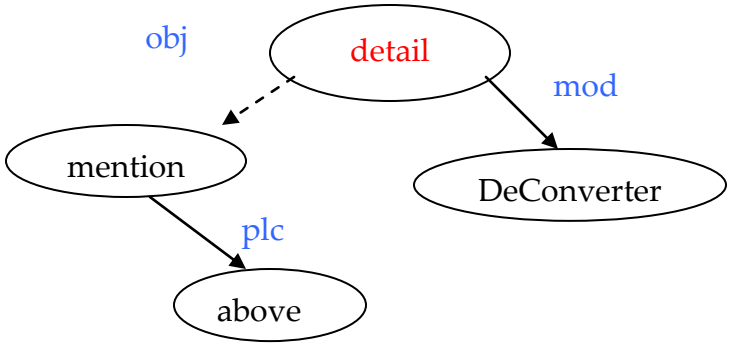

Figure 5.17: Modified UNL graph for a node having untraversed parent

After starting the traversal of UNL graph (Figure 5.16) from entry node, *i.e.*, *'detail'*, system encounters a situation of untraversed parent. The system modifies the UNL graph according to PseudoCode 5.3 and sets *'mention'* node as an active node. It has one untraversed child node, *i.e.*, *'above'*. The *'above'* node has no further unprocessed child so its Punjabi word attribute will be appended to final string used to store generated output. It has parent node *'mention'* which is already traversed and has no unprocessed child node, thus it will be processed by the system and its Punjabi word attribute will be appended to the final string. The *'mention'* node has one virtual parent, *i.e.*, *'detail'* node, hence it will become an active node. This node has one child node, *i.e.*, *'DeConverter'*, so it will become an active node. It has no unprocessed child, so it will be processed and its Punjabi word attribute will be appended to final string. The parent of the *'DeConverter'* node, *i.e.*, *'detail'* node, will now be set as an active node. It has no parent and no unprocessed child, so *'detail'* node is processed by the system and its Punjabi word attribute will be appended to the final string. Since, all the nodes are processed by the system, the control will exit and final output according to syntax plan will be available in the final string.

The output given in (5.74) indicates the final syntax plan of UWs (without constraints) and the output given in (5.75) shows the generated Punjabi sentence after application of morphology, function word insertion and syntax planning phases of the Punjabi DeConverter.

above mention DeConverter detail …(5.74)

ਉੱਪਰ ਦਿੱਤੀ ਡੀਕੰਨਵਰਟਰ ਦੀ ਵਿਆਖਿਆ । …(5.75)

uppar dittī ḍīknnavraṭar dī viākhiā .

5.9.6 Handling of multiple parents

Normally, in a UNL graph each child node has only one parent, *i.e.*, one node plays only a single role. But sometimes, one child may have more than one parent, *i.e.*, one node plays more than one role in a UNL expression. In such a case, the system may encounter a situation where parent of a node may be untraversed or unvisited. This situation is illustrated with an example sentence given in (5.76).

His eyes are affected by strong viral infection. ...(5.76)

The UNL expression for example sentence given in (5.76) is,

{unl}

obj(affect.@present.@entry, eye.@pl)

pos(eye.@pl, he)

agt(affect.@present.@entry, infection)

aoj(viral, infection)

mod(viral, strong)

{/unl} …(5.77)

The UNL graph of UNL expression (5.77) is depicted in Figure 5.18.

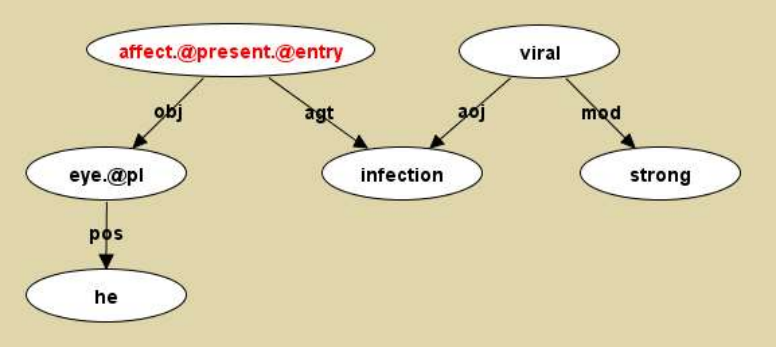

Figure 5.18: UNL graph having multiple parents for UNL expression (5.77)

Here, the entry node *'affect'* has two children *'eye'* and *'infection'* associated with UNL relations *'obj'* and *'agt'*, respectively. The *'agt'* relation is of highest priority, so *'infection'* node will be traversed first by the system. Now, system encounters two parents for an active node *'infection'*. The following strategy has been implemented for handling multiple parents.

If system encounters a node having multiple parents, then its untraversed parent will be detected by the system and the active node will be set as a virtual parent of the untraversed parent. The untraversed parent will be removed as the parent of the active node and the active node will be removed as child of untraversed parent. The untraversed parent node will be set as the active node and further processing of the syntax plan will be carried out according to PseudoCode 5.2.

PseudoCode 5.4 has been used to implement this strategy.

PseudoCode 5.4: Handling nodes with multiple parents nodes

if (node has multiple parents)

    find an untraversed parent;

    set active node as virtual parent of its untraversed parent;

remove untraversed parent from active node's parent list;

remove active node as child of its untraversed parent;

set untraversed parent node as active node;

end-if

PseudoCode 5.4 is inserted between lines numbered 29 and 30 of PseudoCode 5.2 for extending it to handle the UNL graphs having nodes with multiple parents.

According to PseudoCode 5.4, for processing of UNL graph depicted in Figure 5.18, the system will find the untraversed parent of active node having multiple parents, *i.e.*, *'infection'*. The node *'viral'* will be identified as the untraversed parent node of the active node. According to PseudoCode 5.4, the system will set *'infection'* as the virtual parent of the untraversed parent node, *i.e.*, *'viral'* as shown in Figure 5.19.

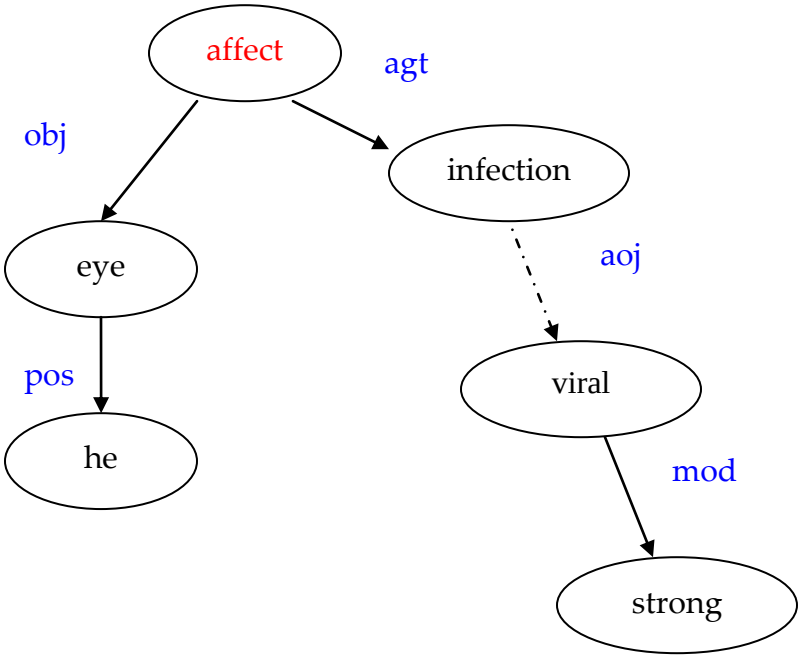

Figure 5.19: Modified UNL graph for a node having multiple parents

Thus, during its syntax plan, the system removes the *'viral'* node as the parent of *'infection'* node and removes *'infection'* as a child of *'viral'* node. The system sets *'viral'* node as an active node after visting *'affect'* and *'infection'* nodes. The *'viral'* node has one child node, *i.e.*, *'strong'*. Thus, it will be processed and its Punjabi word attribute will be appended to final string used to store generated output. Now, the parent of *'strong'*, *i.e.*, *'viral'* node will be set as an active node. Since, all the child nodes of *'viral'* node are processed, so *'viral'* node will be processed and its Punjabi word attribute will be

appended to final string. Now, the virtual parent of *'viral'* node, *i.e.*, *'infection'* node will become an active node. It has no unprocessed child so it will be processed and its Punjabi word attribute will be appended to final string. The parent of the *'infection'* node, *i.e.*, *'affect'* node will now be set as an active node. It has one unprocessed child node, *i.e.*, *'eye'*. So, it will be traversed and become an active node. It has one untraversed child, *i.e.*, *'he'*, so it will be processed and its Punjabi word attribute will be appended to the final string. The parent of *'he'* node, *i.e.*, *'eye'* node will be processed next and its Punjabi word attribute will be appended to the final string. Finally, the entry node, *i.e.*, *'affect'* is processed by the system and its Punjabi word attribute will be appended into the final string.

The output given in (5.78) contains the final syntax plan of UWs (without constraints) and the output given in (5.79) contains the generated Punjabi sentence after application of morphology, function word insertion and syntax planning phases of the Punjabi DeConverter.

strong viral infection he eye affect …(5.78)

ਤਕੜੇ ਜੀਵਾਣਿਕ ਛੂਤ ਨੇ ਉਸ ਦੀਆਂ ਅੱਖਾਂ ਨੂੰ ਅਸਰ ਕੀਤਾ । …(5.79)

takṛē jīvāṇik saṅkramaṇ nē us dīāṃ akkhāṃ nūṃ asar kītā.

5.9.7 Special cases in syntax planning

Dwivedi (2002) reported the shortcomings of syntax planning approach using matrix based priority of relations. In matrix based priority of relations, the priorities are assigned to relations of children nodes sharing a common parent. This approach does not consider the lexical properties of the children nodes. The need for considering the lexical properties of children nodes has been illustrated with the help of example phrases given in (5.80) and (5.82).

Example phrase 1: three days of the conference …(5.80)

UNL expression for this example phrase is,

{unl}

mod(day.@pl, conference)

qua(day.@pl, 3)

{/unl} …(5.81)

The UNL graph of UNL expression (5.81) is depicted in Figure 5.20.

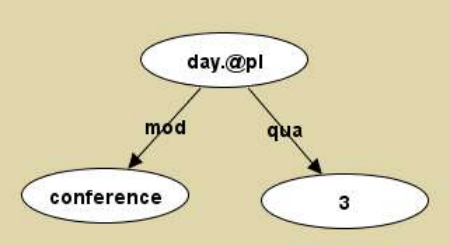

Figure 5.20: UNL graph for UNL expression (5.81)

Example phrase 2: two neighboring nations …(5.82)

UNL expression for this example phrase is,

{unl}

mod(nation.@pl, neighbor)

qua(nation.@pl, 2)

{/unl} …(5.83)

The UNL graph of UNL expression (5.83) is depicted in Figure 5.21.

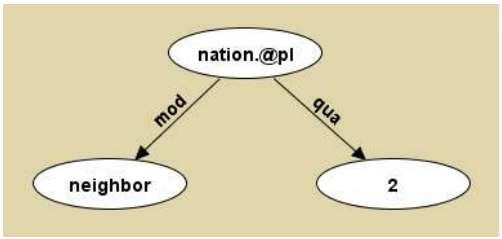

Figure 5.21: UNL graph for UNL expression (5.83)

Punjabi equivalent of first phrase is 'ਕਾਨਫਰੰਸ ਦੇ ਤਿੰਨ ਦਿਨ' *kānpharaṃs dē tinn din'*, which indicate that child node of *'mod'* relation should be traversed first during its syntax planning. It means that *'mod'* relation should have higher priority than *'qua'* relation in the priority matrix. The Punjabi equivalent of second phrase is 'ਦੋ ਗੁਆਂਢੀ ਮੁਲਕ' '*dō guāṇḍhī mulak'*, which indicates that child node of *'qua'* relation should be traversed first during its syntax planning. It means that *'qua'* relation should have higher priority than *'mod'* relation in the priority matrix. These two priorities contradict each other when used in the creation of priority matrix. This change of sequence of insertion happens because *'neighboring'* is an adjective which modifies the noun *'nation'* whereas *'conference'* is a noun indicating something about other noun *'day'*. It means that child of *'mod'* relation will be traversed first if it is a noun but it will be traversed later if the child

is an adjective. These types of special cases have been handled by implementing specific rules in the present work.

Vachhani (2006) and Nalawade (2007) have identified some UNL relations which need to be treated as special cases during their syntax planning for Hindi language. It has been explored that these cases are also applicable for Punjabi language. These UNL relations are as follows.

- '*and'* and/or '*or'* relation(s)
- '*fmt'* relation
- '*cnt'* relation
- *'seq'* relation

Following strategies have been formulated for syntax planning of these UNL relations.

5.9.7.1 Strategy for '*and'* and/or '*or'* relation(s)

For the syntax planning of UNL graph having relation *'and'* relation, PseudoCode 5.2 requires a minor modification, which is illustrated with the help of an example sentence given in (5.84).

I will go to temple and pray for you. …(5.84)

UNL expression for this example sentence is,

{unl}

agt(go.@future.@entry, I(icl>person))

plc(go.@future.@entry, temple)

and(go.@future.@entry, pray)

ben(pray, you)

{/unl} …(5.85)

UNL graph for UNL expression given in (5.85) is depicted in Figure 5.22.

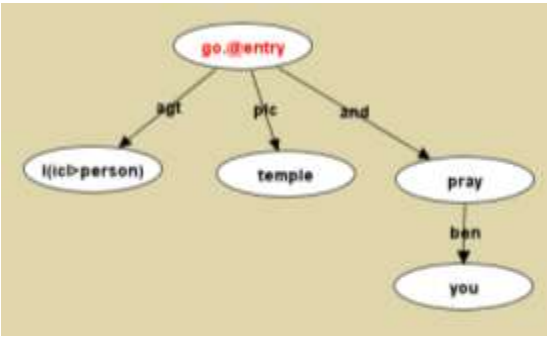

Figure 5.22: UNL graph having relation *'and'* for UNL expression (5.85)

According to PseudoCode 5.2, the syntax planning of UNL graph given in Figure 5.22 will result into the syntax planning of UWs (without constraints) as given in (5.86).

I temple you pray go …(5.86)

This word order is not correct for Punjabi language, because the position of UW *'go'* is wrong. The correct syntax plan of UWs according to Punjabi language is,

I temple go you pray …(5.87)

In order to generate the correct syntax plan for a UNL graph having *'and'* relation, following strategy has been implemented.

If system encounters relation *'and'* as the highest priority relation, then parent node of relation *'and'* will be processed first and its child node will be set as an active node. Further syntax planning will be carried out according to PseudoCode 5.2.

For the syntax planning of UNL graph depicted in Figure 5.22, the syntax plan starts from entry node *'go'* having three children with *'agt'*, *'plc'* and *'and'* relations. According to priority matrix, the *'agt'* relation is of the highest priority, so the node *'I(icl>person)'* will be processed first followed by node *'temple'* with *'plc'* relation. After this, the system will traverse the node with *'and'* relation. According to aforementioned strategy, the system will first process the parent node of relation *'and'* and its Punjabi word attribute will be appended to final string. Now, the child node of *'and'* relation, *i.e.*, *'pray'* node will become an active node. It will result into the processing of *'you'* node followed by *'pray'* node. The syntax plan of UWs (without constraints) is given in (5.88). Thus, the Punjabi sentence given in (5.89) will be generated after application of morphology, function word insertion and syntax planning phases of the Punjabi DeConverter.

I temple go you pray ...(5.88)

ਮੈਂ ਮੰਦਰ ਜਾਵਾਂਗਾ ਅਤੇ ਤੇਰੇ ਲਈ ਪ੍ਰਾਰਥਨਾ ਕਰਾਂਗਾ । …(5.89)

maiṃ mandar jāvāṅgā atē tērē laī prārthanā karāṅgā .

The UNL graph consisting of '*or'* relation has also been handled by the same strategy that is adopted for *'and'* relation. This is illustrated with the help of an example sentence given in (5.90).

Plant 2 or 3 trees in different directions. …(5.90)

UNL expression for this example sentence is given in (5.91) and corresponding UNL graph is given in Figure 5.23.

{unl}

obj(plant.@entry, tree.@pl)

scn(plant.@entry, direction.@pl)

mod(direction.@pl, different)

qua(tree.@pl, :01)

or:01(2.@entry, 3)

{/unl} …(5.91)

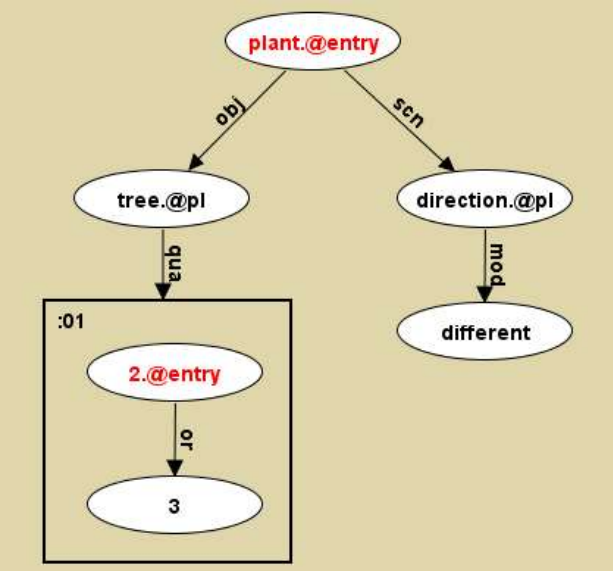

Figure 5.23: UNL graph having relation *'or'* for UNL expression (5.91)

As mentioned above, the strategy adopted for *'and'* relation has also been used for *'or'* relation, accordingly, in Figure 5.23 the parent of *'or'* relation, *i.e.*, *'2'* will be processed before its child node *'3'*. Since, the relation *'obj'* has higher priority than relation *'scn'* in the priority matrix, the syntax plan of UWs (without constraints) will be obtained as given in (5.92). The Punjabi output generated after application of morphology, function word insertion and syntax planning phases is given in (5.93).

2 3 tree different direction plant …(5.92)

2 3 ਦਰਖ਼ਤ ਅਲੱਗ ਦਿਸ਼ਾਵਾਂ ਵਿਚ ਉਗਾਓ । …(5.93)

2 3 dar<u>kh</u>at alagg dishāvāṃ vic ugāō .

This strategy for syntax planning of UNL graph having *'and'* and/or *'or'* relations has

been implemented and the PseudoCode 5.5 contains the instructions corresponding to its implementation.

PseudoCode 5.5: Handling of *'and'* and/or *'or'* relations

if (highest priority relation is *'and'* or *'or'*)

process active node by appending its Punjabi word attribute into final string;

mark the node as special node to indicate that it is already processed;

end-if

PseudoCode 5.5 is inserted between lines 13 and 14; and also between lines 26 and 27 of PseudoCode 5.2 for extending it to handle the UNL graphs having *'and'* and/or *'or'* relations.

5.9.7.2 Strategy for *'fmt'* relation

Again, for the syntax planning of UNL graph having *'fmt'* relation, PseudoCode 5.2 requires slight modification, as illustrated in the following example sentence.

I can learn the list from a to z. …(5.94)

UNL expression for this example sentence is given in (5.95) and corresponding UNL graph is given in Figure 5.24.

{unl}
agt(learn(icl>do).@entry.@ability, I(icl>person))
obj(learn(icl>do).@entry.@ability, list.@def)
man(learn(icl>do).@entry.@ability, a(icl>letter))
fmt(a(icl>letter), z(icl>letter))
{/unl} ...(5.95)

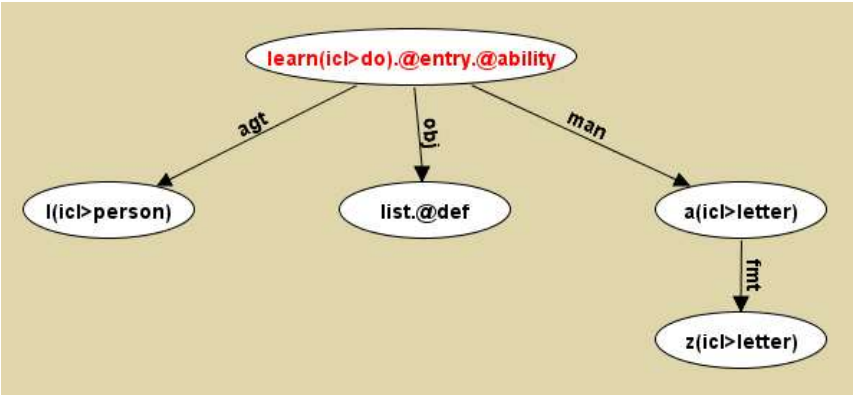

Figure 5.24: UNL graph having relation *'fmt'* for UNL expression (5.95)

For UNL graph depicted in Figure 5.24, the PseudoCode 5.2 will generate the syntax plan of UWs (without constraints) as shown in (5.96).

I z a list learn ...(5.96)

Here, *'z'* appears before *'a'*, which is incorrect word order according to Punjabi language; *'z'* should appear after *'a'*. In order to generate correct word order, a different strategy needs to be followed for *'fmt'* relation. Following strategy has been implemented for syntax plan of *'fmt'* relation.

If system encounters relation *'fmt'* as the highest priority relation, the parent node of relation '*fmt'* is processed first followed by the processing of its child node. After the processing of child node, the child node will be set as an active node. Further processing will be carried out according to PseudoCode 5.2.

PseudoCode 5.6 has been used to implement this strategy.

PseudoCode 5.6: Handling of *'fmt'* relation

if (highest priority relation is *'fmt'*)

process active node by appending its Punjabi word attribute into final string;
mark the node as special node to indicate that it is already processed;
process its child node by appending its Punjabi word attribute into final string;
mark the child node as special node to indicate that it is already processed;
set the child node as the active node;

end-if

PseudoCode 5.6 is inserted between lines 13 and 14; and also between 26 and 27 of PseudoCode 5.2 for extending it to handle the UNL graphs having *'fmt'* relation.

The syntax plan of UWs (without constraints) for (5.96) will be generated as given in (5.97) by PseudoCode 5.6. The Punjabi output after application of morphology, function word insertion and syntax planning phases is given in (5.98).

I a z list learn ...(5.97)

ਮੈਂ a ਤੋਂ z ਤੱਕ ਲੜੀ ਯਾਦ ਕੀਤੀ । …(5.98)

maiṃ a tōṃ z takk laṛī yād kītī .

5.9.7.3 Strategy for '*cnt'* relation

In case of UNL graph having *'cnt'* relation, Punjabi word attribute of parent node should be appended at leftmost position in the generated output. This concept is illustrated with the help of an example sentence given in (5.99).

A language generator "DeConverter" is necessary. …(5.99)

UNL expression for this example sentence is given in (5.100) and corresponding UNL graph is given in Figure 5.25.

{unl}

cnt(language generator(icl>tool), deconverter(icl>tool).@double_quote)

aoj(necessary.@present.@entry, language generator(icl>tool))

{/unl} …(5.100)

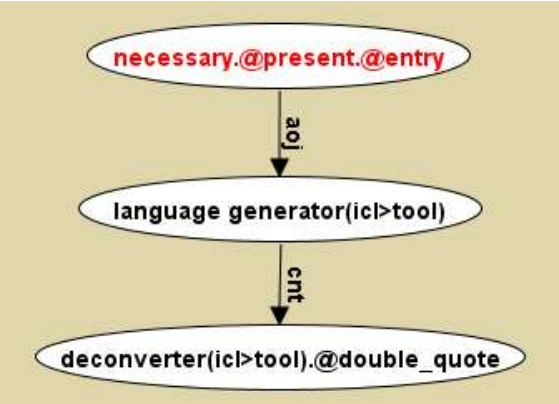

Figure 5.25: UNL graph having *'cnt'* relation for UNL expression (5.100)

If we follow PseudoCode 5.2, syntax plan of UWs (without constraints) for this UNL graph will be as given in (5.101) and Punjabi sentence generated from this syntax plan is given in (5.102).

deconverter language generator necessary …(5.101)

"ਡੀਕੰਨਵਰਟਰ" ਭਾਸ਼ਾ ਜੰਨਰੇਟਰ ਜ਼ਰੂਰੀ ਹੈ। …(5.102)

ḍīknnavraṭar bhāshā jannrēṭar zarūrī hai .

This is not a correct syntax plan and as a result an incorrect translation. In order to produce correct syntax plan for Punjabi language, Punjabi word attribute of parent node of *'cnt'* relation, *i.e.*, 'ਭਾਸ਼ਾ ਜੰਨਰੇਟਰ' '*bhāshā jannrēṭar'* should be appended at leftmost position in the generated output. Following this strategy, Punjabi output given in (5.103)

will be generated after application of morphology, function word insertion and syntax planning phases.

ਭਾਸ਼ਾ ਜੰਨਰੇਟਰ **"**ਡੀਕੰਨਵਰਟਰ**"** ਜ਼ਰੂਰੀ ਹੈ । …(5.103)

bhāshā jannrēṭar ḍiknnavraṭar zarūrī hai .

The above discussed strategy for *'cnt'* relation has been implemented in PseudoCode 5.7.

PseudoCode 5.7: Handling of *'cnt'* relation

if (highest priority relation is *'cnt'*)

process the active node by appending its Punjabi word attribute to the left most position in the generated output;

mark the node as special node to indicate that it is already processed;

end-if

PseudoCode 5.7 is inserted between lines numbered 13 and 14; and also between lines numbered 26 and 27 of PseudoCode 5.2 for extending it to handle the UNL graphs having *'cnt'* relation.

5.9.7.4 Strategy for *'seq'* relation

The *'seq'* relation indicates a sequence of events in the sentence. It results into insertion of ਪਹਿਲਾਂ *pahilāṃ 'before'* or ਬਾਅਦ *bāad 'after'* in the generated Punjabi sentence. When there is a sequence of events, one event refers to the other event. The *'@reference'* has been used as UNL attribute to resolve the referring event as suggested by Nalawade (2007). This attribute may appear with parent or with child node of the *'seq'* relation to specify the referring event.

Following strategy has been implemented for syntax planning of UNL graph having *'seq'* relation.

(i) If parent node of *'seq'* relation has *'@reference'* attribute, then child node of *'seq'* relation should be considered as lowest priority node and it should be processed after its parent node.

(ii) If child node of *'seq'* relation has *'@reference'* attribute, then child node of *'seq'* relation should be considered as highest priority node and it should be traversed first.

In order to implement this strategy for *'seq'* relation, the priority of relation *'seq'* has been set to minimum out of all the relations in the priority matrix '*M'*. Step (i) of this strategy is illustrated below with the help of an example sentence, given in (5.104).

Before this chair there was a table in the room. …(5.104)

UNL expression for this example sentence is given in (5.105) and corresponding UNL graph is given in Figure 5.26.

{unl}

seq(chair.@reference.@entry, exist.@past)

mod(chair.@reference.@entry, this)

plc(exist.@past, room.@def)

aoj(exist.@past, table)

{/unl} ...(5.105)

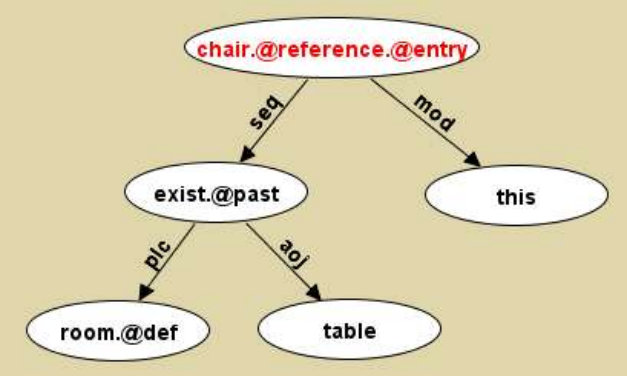

Figure 5.26: UNL graph having *'seq'* relation and parent with *'@reference'* attribute

For the syntax planning of UNL graph given in Figure 5.26, system will start traversing from entry node *'chair'*. This node has *'@reference'* attribute. Thus, the system will check the relations of children of entry node and will find that one of its child has *'seq'* relation, the priority of *'seq'* is lowest in the priority matrix and the system will traverse *'this'* node first. Since, it does not have a child; its Punjabi word attribute will be appended to the final string used to store generated output. The parent of *'this'* node, *i.e.*, *'chair'* will now be set as an active node. It has only one unprocessed child node with *'seq'* relation. Since, the parent node has *'@reference'* attribute, thus, according to step (i) of the strategy, the parent of *'seq'* relation will be processed first and then the child node will be traversed. The Punjabi word attribute of parent node, *i.e.*, *'chair'* node will be appended to the final string. The system now traverses the *'exist'* node. It has two children, since the priority of relation '*plc'* is higher than relation *'aoj'* in the priority matrix, it will result into the processing of *'room'* node followed by *'table'* node. Now, the *'exist'* will become an active node. Since, all the children of *'exist'* node have been

processed, so, finally *'exist'* node will be processed and syntax planning process will be completed.

Output given in (5.106) contains the final syntax plan of UWs (without constraints) and the output given in (5.107) contains the generated Punjabi sentence after application of morphology, function word insertion and syntax planning phases of the Punjabi DeConverter.

this chair room table exist …(5.106)

ਇਸ ਕੁਰਸੀ ਤੋਂ ਪਹਿਲਾਂ ਕਮਰੇ ਵਿਚ ਮੇਜ਼ ਮੌਜੂਦ ਸੀ । …(5.107)

is kurasī tōṃ pahilāṃ kamrē vic mēz maujūd sī.

The requirement of step (ii) of the proposed strategy is now illustrated with the help of an example sentence, given in (5.108).

Urea should be mixed with water and the mixture should be sprayed twice. …(5.108)

UNL expression for this example sentence is given in (5.109) and corresponding UNL graph is given in Figure 5.27.

{unl}
man(spray.@entry.@should, time.@pl)
qua(time.@pl, two)
seq(spray.@entry.@should, mix.@reference.@should)
cob(mix.@reference.@should, water)
obj(mix.@reference.@should, urea)
mod(spray.@entry.@should, mixture)
{/unl} …(5.109)

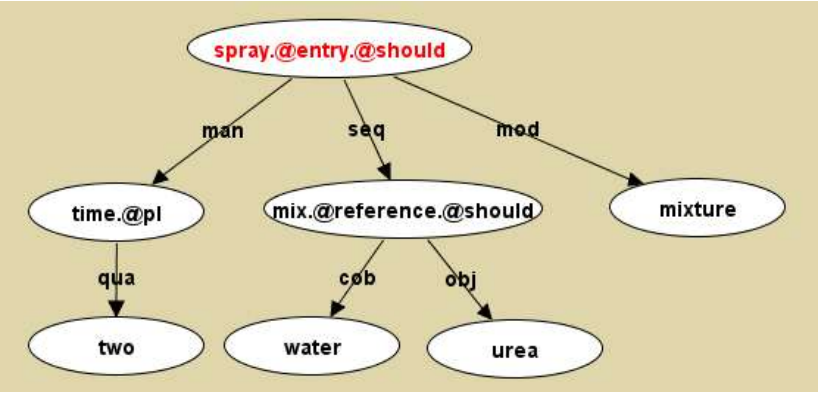

Figure 5.27: UNL graph having *'seq'* relation and child node with *'@reference'* attribute

For the syntax planning of UNL graph given in Figure 5.27, system will start traversing from entry node *'spray'*. The system will check the relations of children of entry node and will find that one of its children has *'seq'* relation with *'@reference'* attribute, so the priority of this relation will be set as highest as per step (ii) of the strategy. So, the system will traverse the *'mix'* node first. This node has two children with relations *'cob'* and *'obj'*. Since the priority of relation *'cob'* is higher than *'obj'* in the priority matrix, it will result into the processing of node *'water'* followed by *'urea'* node and then *'mix'* node. After processing of *'mix'* node, the *'spray'* node will become an active node. It has two unprocessed children with relation *'man'* and relation *'mod'*. Since, the priority of relation *'mod'* is higher than relation *'man'*, the system will process the *'mixture'* node first followed by *'two'* node and then *'time'* node. After processing *'time'* node, the *'spray'* node will become active node. Since, all the children of *'spray'* node have been processed, now *'spray'* node will be processed and the system will exit out of syntax planning process.

The output given in (5.110) contains the final syntax plan of UWs (without constraints) and the output given in (5.111) contains the generated Punjabi sentence after application of morphology, function word insertion and syntax planning phases of the Punjabi DeConverter.

water urea mix mixture two time spray …(5.110)

ਪਾਣੀ ਵਿਚ ਯੂਰੀਆ ਮਿਲਾਣ ਤੋਂ ਬਾਅਦ ਮਿਸ਼ਰਨ 2 ਵਾਰ ਛਿੜਕੋ । …(5.111)

pāṇī vicc yūrīā milāṇ tōṃ bāad mishran 2 vār chiṛkō .

The PseudoCode 5.8 and PseudoCode 5.9 have been implemented for the syntax plan of *'seq'* relation.

PseudoCode 5.8: Implementation of step (i) of strategy for *'seq'* relation

if (highest priority relation is *'seq'* and active node has *'@reference'* attribute)

process the active node by appending its Punjabi word attribute into the generated output;

mark the node as special node to indicate that it is already processed;

end-if

PseudoCode 5.9: Implementation of step (ii) of strategy of *'seq'* relation

if (active node has a '*seq'* relation as one of its child relation and it does not has *'@reference'* attribute)

set *'seq'* as a highest priority relation;

end-if

PseudoCode 5.8 is inserted between lines numbered 13 and 14; and also between lines numbered 26 and 27 of PseudoCode 5.2. PseudoCode 5.9 is inserted between lines numbered 12 and 13; and also between lines numbered 25 and 26 of PseudoCode 5.2. These insertions allow PseudoCode 5.2 to handle the UNL graphs with *'seq'* relation.

5.9.8 Syntax planning for clausal sentences

A sentence which contains one independent clause and one or more dependent (or subordinate) clauses is called a complex sentence. There are three basic types of dependent clauses, namely, noun clause, adjective clause and adverb clause. In this section, syntax planning of clausal sentences has been presented.

5.9.8.1 Syntax planning of sentences with noun clause

A noun clause is identified in a sentence, if the object of the main verb is a complete meaningful sentence in itself. In UNL expression, object is represented by a scope node and it has a proper subject/predicate structure (Dwivedi, 2002). Let us illustrate this concept with the help of an example sentence given in (5.112).

Ram said that Sukhwinder is good. …(5.112)

UNL expression for this example sentence is given in (5.113) and corresponding UNL graph is given in Figure 5.28.

{unl}
agt(say.@entry.@past, Ram(icl>person))
obj(say.@entry.@past, :01)
aoj:01(good.@entry.@present, Sukhwinder(icl>person))
{/unl} …(5.113)

Here, scope node ':01' acts as the object of the main clause and is used to represent *'Sukhwinder is good'* which is a complete meaningful sentence in itself.

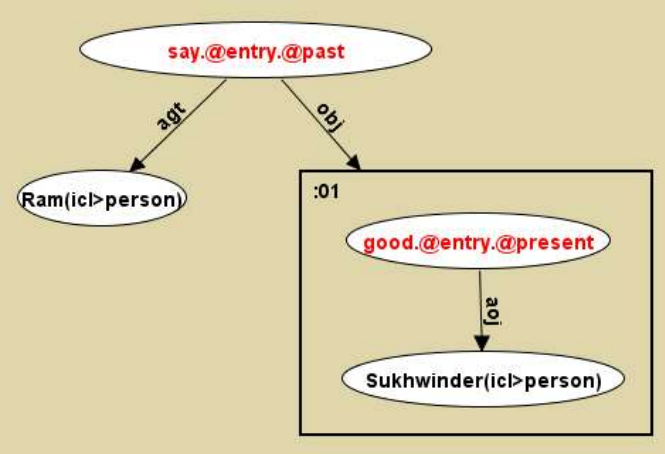

Figure 5.28: UNL graph for noun clause sentence given in (5.112)

UNL expression given in (5.113) involves a scope node, so algorithm 5.4 has been used to produce the syntax plan for this UNL graph with a scope node. The syntax planning produced by this algorithm considering UWs (without constraints) and scope node as a single node is given in (5.114) and its complete syntax planning is given in (5.115).

Ram :01 say …(5.114)

Ram Sukhwinder good say …(5.115)

As such, we are not able to produce the correct syntax plan, as given in (5.116) using this algorithm.

Ram say Sukhwinder good …(5.116)

In order to generate the correct word order, a different strategy needs to be followed for syntax planning of noun clause sentences. Following strategy has been implemented for syntax planning of these types of sentences.

While traversing a UNL graph, if the system encounters relation *'obj'* as highest priority relation having a scope node as a child of this relation, then system will check the presence of subject/predicate structure in the scope node. If a scope node has subject/predicate structure, then scope node represents a complete sentence. In this case, the system will first process the parent of *'obj'* relation and its Punjabi word attribute will be appended into generated output. The child scope node will be set as active node and further syntax planning will be performed using algorithm 5.4.

Algorithm 5.4 has been modified to handle the noun clause sentences, and the modified algorithm is given below.

Algorithm 5.5: Processing of noun clause sentences

(i) Check the presence of subject/predicate structure in the scope node. If a scope node has subject/predicate structure, then set clausal flag to one.

(ii) Develop the syntax plan by considering scope node as a single node and applying extended PseudoCode 5.2 (to process simple UNL graph, untraversed parent, multiple parents and all special cases in the syntax planning of a UNL graph).

   a. During processing with extended PseudoCode 5.2, if system encounters relation *'obj'* as highest priority relation having a scope node as a child of this relation and clausal flag as set, then system will first process the parent of *'obj'* relation and its Punjabi word attribute will be appended into final string. The system will mark the node as special node to indicate that it is already processed and its child node will be set as the active node.

(iii) Develop the syntax plan of scope node's UNL graph using extended PseudoCode 5.2.

(iv) Replace the scope node in the output generated in step (ii) with the output generated in step (iii) in order to get final generated sentence.

Following this strategy, the syntax planning of UWs (without constraints) by considering scope node as a single node of UNL expression (5.113) is given in (5.117) and complete syntax plan is given in (5.118). The Punjabi output generated after application of morphology, function word insertion and syntax planning phases is given in (5.119).

Ram say :01 …(5.117)

Ram says Sukhwinder good …(5.118)

ਰਾਮ ਨੇ ਕਿਹਾ ਕਿ ਸੁੱਖਵਿੰਦਰ ਚੰਗਾ ਹੈ । …(5.119)

rām nē kihā ki sukkhvindar caṅgā hai.

5.9.8.2 Syntax planning of sentences with adjective clause

An adjective clause can be 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). In a UNL graph, adjective clause is present if a scope node containing a subject/predicate structure has an *'aoj'* relation with the entry node of the main UNL graph and the entry node of scope node's UNL graph acts as child node of relation *'obj'*.

This concept has been illustrated with the help of an example sentence given in (5.120).

The child that Ram saved was beautiful. …(5.120)

UNL expression for this example sentence is given in (5.121) and corresponding UNL graph is given in Figure 5.29.

{unl}

aoj(beautiful.@present.@entry, :01)

obj:01(save.@past.@complete, child.@entry)

agt:01(save.@past.@complete, Ram(icl>person))

{/unl} …(5.121)

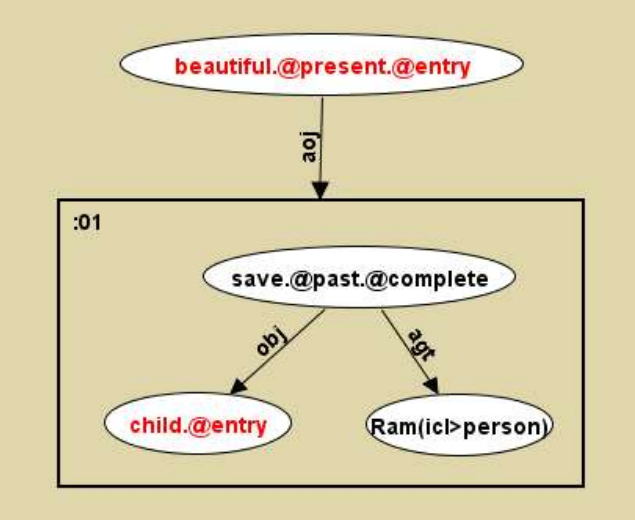

Figure 5.29: UNL graph for adjective clause sentence given in (5.120)

Here, scope node ':01' has an *'aoj'* relation with the entry node of main UNL graph and entry node of scope node's UNL graph, *i.e.*, *'child'* acts as the child node of relation *'obj'* for parent node, *i.e.*, *'save'*. It indicates that example sentence given (5.120) is an adjective clause sentence and asserts that in the sentence central focus is on the *'child'* and not the action *'save'*.

During the syntax plan of scope node, the system will encounter the situation of untraversed parent node. Accordingly, the system will generate the syntax plan of UWs (without constraints) as given in (5.122).

Ram save child …(5.122)

Following algorithm 5.5, the complete syntax plan of UWs (without constraints) by considering scope node as a single node will be as given in (5.123) and the complete syntax plan will be as given in (5.124).

:01 beautiful …(5.123)

Ram save child beautiful …(5.124)

As such, we could not get a correct syntax plan fit for Punjabi language structure. The correct syntax plan of UWs is given in (5.125) and equivalent Punjabi translated sentence is given in (5.126).

child Ram save beautiful …(5.125)

ਜਿਸ ਬੱਚੇ ਨੂੰ ਰਾਮ ਨੇ ਬਚਾਇਆ ਉਹ ਸੋਹਣਾ ਸੀ। …(5.126)

jis baccē nūṃ rām nē bacāiā uh sōhṇā sī.

For generating the correct word order, there is a need to make the changes in the syntax planning process. It has also been noted that in case of generation of adjective clause sentences, Punjabi uses ਜੋ *jō* and ਉਹ *uh* structure. It means that in the process of generation of adjective clause Punjabi sentence, a form of ਜੋ *jō* followed by the subordinate clause, followed by a corresponding form of ਉਹ *uh* and the main clause are to be used.

Following strategy has been implemented to produce correct word order for syntax plan of UNL graph having an adjective clause.

While traversing a UNL graph, if system encounters an *'aoj'* relation as highest priority relation with scope node as a child node and clausal flag as set, then system will check the entry node of scope node's UNL graph. If it has an untraversed parent with *'obj'* relation, then the system will traverse the untraversed parent first, by considering child node of *'obj'* relation as the highest priority relation and further syntax planning process will be carried out according to algorithm 5.5.

Algorithm 5.5 for handling of syntax planning of noun clause sentences has been extended to algorithm 5.6 to handle both noun and adjective clause sentences.

Algorithm 5.6: Handling of noun and adjective clause sentences

(i) Check the presence of subject/predicate structure in the scope node. If a scope node has subject/predicate structure, then set clausal flag to one.

(ii) Develop the syntax plan by considering scope node as a single node and applying extended PseudoCode 5.2 (to process simple UNL graph, untraversed parent, multiple parents and all special cases in the syntax planning of a UNL graph).

a. During processing with extended PseudoCode 5.2, if system encounters relation *'obj'* as highest priority relation having a scope node as a child of this relation and clausal flag as set, then system will first process the parent of *'obj'* relation and its Punjabi word attribute will be appended into final string. The system will mark the node as special node to indicate that it is already processed and its child node will be set as the active node.

b. During processing with extended PseudoCode 5.2, if system encounters relation *'aoj'* as highest priority relation having a scope node as a child of this relation with clausal flag as set, then set adjective flag to one.

(iii) Develop the syntax plan of scope node's UNL graph using extended PseudoCode 5.2. If during its syntax plan, system encounters an entry node having an untraversed parent with *'obj'* relation with adjective flag as set, then the system will traverse the untraversed parent first, by considering child node of *'obj'* relation as highest priority relation.

(iv) Replace the scope node in the output generated in step (ii) with the output generated in step (iii) in order to get final generated sentence.

(v) If adjective flag is set then append the appropriate form of ਜੋ *jō* and ਉਹ *uh* structure in the generated output.

Following this algorithm, during syntax planning of scope node's UNL graph given in Figure 5.29, the system will first traverse the untraversed parent of node *'child'*, *i.e.*, *'save'*. It has two children, one with *'obj'* relation and other with *'agt'* relation. The *'obj'* relation has highest priority and system will traverse this first followed by *'agt'* relation. Thus, final syntax plan of scope node will be as given in (5.127).

child Ram save …(5.127)

The complete syntax plan for example sentence given in (5.120) will be as given in (5.128). The Punjabi output generated after application of morphology, function word insertion and syntax planning phases is given in (5.129).

child Ram save beautiful …(5.128)

ਜਿਸ ਬੱਚੇ ਨੂੰ ਰਾਮ ਨੇ ਬਚਾਇਆ ਉਹ ਸੋਹਣਾ ਸੀ । …(5.129)

jis baccē nūṃ rām nē bacāiā oh sōhṇā sī.

In the generated Punjabi sentence given in (5.129), ਜਿਸ *jis* has been used as the form of ਜੋ *jō* followed by the subordinate clause 'ਬੱਚੇ ਨੂੰ ਰਾਮ ਨੇ ਬਚਾਇਆ' *'baccē nūṃ rām nē bacāiā'*, followed by ਉਹ *oh* and main clause 'ਸੋਹਣਾ ਸੀ' *sōhṇā sī.*.

5.9.8.3 Syntax planning of sentences with adverb clause

Adverb clause is the subordinate clause that acts as an adverb in a sentence. It may modify some verb, adjective or adverb in the main clause. The adverb clauses show relationship for time, for condition, for place and for manner with the main clause (Giri, 2000). The concept of adverb clause of time is illustrated below with the help of an example sentence given in (5.130).

When I and Sukhwinder will go to Tokyo, then we will see big tower. …(5.130)

UNL expression for this example sentence is given in (5.131) and corresponding UNL graph is given in Figure 5.30.

{unl}
and(I(icl<person), Sukhwinder(icl<person))
plt(go(icl>do).@entry.@future, Tokyo (icl>place))
agt(go(icl>do).@entry.@future, I(icl<person))
mod:01(tower(icl>building), large(aoj>thing))
obj:01(see(icl<event).@future.@pl.@entry, tower(icl>building))
agt:01(see(icl<event).@future.@pl.@entry, I(icl>person).@pl)
tim(go(icl>do).@entry.@future, :01)
{/unl} …(5.131)

Here, scope node ':01' acts as the child of entry node *'go'* with relation *'tim'*. It indicates that it is a case of adverb clause sentence with time modification.

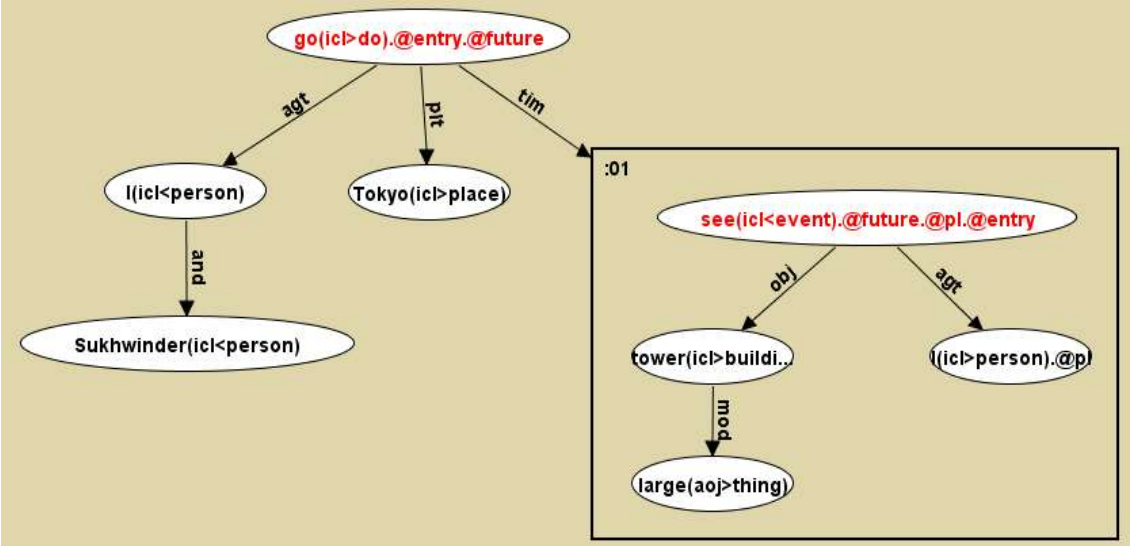

Figure 5.30: UNL graph with adverb clause for time for example sentence (5.131)

According to algorithm 5.6 (UNL graph with scope node and handling of noun and adjective clause) the system will generate the syntax plan of UWs (without constraints) by considering scope node as a single node as given in (5.132) and complete syntax plan will be as given in (5.133).

I Sukhwinder Tokyo :01 go …(5.132)

I Sukhwinder Tokyo we large tower see go …(5.133)

It is not a correct syntax plan of UWs as per Punjabi language structure. The correct syntax plan of UWs is given in (5.134).

I Sukhwinder Tokyo go we large tower see …(5.134)

In order to generate the correct word order, there is a need to make the changes in the syntax planning process. Following strategy has been implemented to produce the correct word order for syntax plan of UNL graph having an adverb clause.

If during traversal of a UNL graph, system encounters relation *'tim' or 'con' or 'plc'* or *'man'* as a highest priority relation having a scope node as a child of this relation and clausal flag as set, then system will process the parent of *'tim' or 'con' or 'plc'* or *'man'* relation and its Punjabi word attribute will be appended to the generated output. The child node, *i.e.*, a scope node will set as the active node and further syntax planning will be performed using algorithm 5.6.

Following this strategy, the system will produce the syntax plan given in (5.135) of UWs (without constraints) by considering scope node as a single node. The complete syntax

plan will be as given in (5.136). The Punjabi output generated after application of morphology, function word insertion and syntax planning phases is given in (5.137).

I Sukhwinder Tokyo go :01 …(5.135)

I Sukhwinder Tokyo go we large tower see …(5.136)

ਜਦੋਂ ਮੈਂ ਅਤੇ ਸੁੱਖਵਿੰਦਰ ਟੋਕੀਓ ਜਾਵਾਂਗੇ ਓਦੋਂ ਅਸੀਂ ਵੱਡਾ ਮਿਨਾਰ ਦੇਖਾਂਗੇ । …(5.137)

jadōṃ maiṃ atē sukkhvindar ṭōkīō jāvāṅgē ōdōṃ asīṃ vaḍḍā minār dēkhāṅgē.

It has also been noted that in case of generation of adverb clause of Punjabi sentences with time relation, ਜਦੋਂ *jadōṃ 'when'* and ਓਦੋਂ *ōdōṃ 'then'* form of structure is used. Similarly, in case of adverb clause for condition ਜੇਕਰ/ਜੇ *jēkar/jē 'if'* and ਤਾਂ/ਤੇ *tāṃ/tē 'then'* form of structure will be inserted into generated output, in case of adverb clause for place ਜਿੱਥੇ *jitthē* and ਉੱਥੇ *utthē* form of structure will be inserted into generated output and in case of adverb clause manner ਜਿੱਦਾਂ *jiddāṃ* and ਓਦਾਂ *ōdāṃ* form of structure will be inserted into generated output.

Algorithm 5.6 has further been extended to form algorithm 5.7, so that it can handle the syntax planning of noun, adjective and adverb clause sentences.

Algorithm 5.7: Handling of noun, adjective and adverb clause sentences

(i) Check the presence of subject/predicate structure in the scope node. If a scope node has subject/predicate structure, then set clausal flag to one.

(ii) Develop the syntax plan by considering scope node as a single node and applying extended PseudoCode 5.2 (to process simple UNL graph, untraversed parent, multiple parents and all special cases in the syntax planning of a UNL graph).

a. During processing with extended PseudoCode 5.2, If system encounters relation *'obj'* as a highest priority relation having a scope node as a child of this relation and clausal flag as set, then system will first process the parent of *'obj'* relation and its Punjabi word attribute will be appended into final string. The system will mark the node as special node to indicate that it is already processed and its child node will be set as the active node.

b. During processing with extended PseudoCode 5.2, if system encounters relation *'aoj'* as a highest priority relation having a scope node as a child of this relation with clausal flag as set, then set adjective flag to one.

c. During processing with PseudoCode 5.2, if system encounters relation *'tim' or 'con' or 'plc'* or *'man'* as a highest priority relation having a scope node as a child of this relation and clausal flag as set, then system will process the parent of *'tim' or 'con' or 'plc'* or *'man'* relation and will append its Punjabi word attribute to the generated string by setting corresponding type of adverb flag to one. The system will mark the node as special node to indicate that it is already processed and its child node will be set as the active node.

(iii) Develop the syntax plan of scope node's UNL graph using extended PseudoCode 5.2. If during its syntax plan, system encounters an entry node having an untraversed parent with *'obj'* relation with adjective flag as set, then the system will traverse the untraversed parent first, by considering child node of *'obj'* relation as a highest priority relation.

(iv) Replace the scope node in the output generated in step (ii) with the output generated in step (iii) in order to get the generated sentence.

(v) If adjective flag is set then append the appropriate form of ਜੋ *jō* and ਉਹ *uh* structure in the generated output.

(vi) If adverb clause for condition flag is set then ਜੇਕਰ/ਜੇ *jēkar/jē 'if'* and ਤਾਂ/ਤੇ *tāṃ/tē 'then'* form of structure will be appended into generated output.

(vii) If adverb clause for place flag is set ਜਿੱਥੇ *jitthē* and ਉੱਥੇ *utthē* form of structure will be appended into generated output.

(viii) If adverb clause for manner flag is set then ਜਿੱਦਾਂ *jiddāṃ* and ਓਦਾਂ *ōdāṃ* form of structure will be appended into generated output.

Algorithm 5.7 will enable the system to handle syntax planning of clausal sentences. All the algorithms and pseudocodes included in this chapter have been implemented in Java to develop the proposed Punjabi DeConverter. This DeConverter forms a part of the web interface developed in this work for Punjabi-UNL EnConversion and UNL-Punjabi DeConversion process.

Chapter Summary

The proposed UNL to Punjabi DeConverter that generates natural language Punjabi sentence from a given UNL expression has been discussed, in detail, in this chapter. Punjabi DeConverter involves UNL parser, lexeme selection, morphology generation, function word insertion and syntax planning phases. First stage of a DeConverter is UNL parser which parses the input UNL expression to build a node-net from the input UNL expression. During lexeme selection phase, Punjabi root words and their dictionary attributes are selected for the given UWs in the input UNL expression from the Punjabi-UW dictionary. After that, the nodes are ready for generation of morphology according to the target language in the morphology phase. The proposed system makes use of morphology rule base for Punjabi language to handle attribute label resolution morphology; relation label resolution morphology; and noun, adjective, pronoun and verb morphology. In function word insertion phase, the function words are inserted to the morphed words. These function words are inserted in the generated sentence based on nine column rule base. Finally, the syntax planning phase is used to define the word order in the generated sentence so that output matches with a natural language sentence. Algorithms and pseudocodes have been implemented for syntax planning for simple sentences, syntax planning of UNL graph with a scope node, untraversed parent, multiple parents, special cases in syntax planning and clausal sentences. All the algorithms and pseudocodes used in different phases of Punjabi DeConverter have been implemented in Java to develop the proposed UNL-Punjabi DeConverter.

Chapter 6
Results and Discussion

6.1 Introduction

This chapter presents the results and discussions on the work carried out in this thesis. This work, as mentioned earlier, has focused on the development of EnConverter and DeConverter for Punjabi language. The development process of these tools has been explained in detail in the previous chapters. Apart from the process that was followed, the outcomes of EnConverter and DeConverter have also been discussed. The important aspect of these tools that has been presented in this chapter is their testing for the perspective of a good machine translation system.

The testing of a MT system is an important aspect in order to establish the usefulness of the system. The evaluation of proposed system, consisting of Punjabi EnConverter and DeConverter, has been performed with the help of one thousand Punjabi sentences. These sentences have been selected in such a way that generation of all possible UNL relations and attributes can be tested. The results of adequacy, fluency tests and BLEU score are very encouraging. The quantitative test, namely, error analysis has also been performed by calculating Sentence Error Rate (SER) and Word Error Rate (WER) of the proposed UNL based MT system. The results of this experimentation are presented in subsequent sections.

6.2 Selection of sentences for test cases

An important aspect in MT system evaluation is to make appropriate selection of sentences for evaluating an MT system. According to Balkan (1998), there are three types of sources for evaluation of MT system, namely, test corpora (a collection of naturally occurring text), test suites (a collection of usually artificially constructed inputs, where each input is designed to probe a system's treatment to a specific phenomenon) and test collections (a set of inputs associated with a corresponding set of expected outputs). Accordingly, one thousand sentences have been considered for evaluation of proposed UNL based MT system for Punjabi language. As mentioned, these sentences have been

considered in such a way that generation of all possible UNL relations and attributes can be tested.

6.3 Evaluation of Punjabi EnConverter

Evaluation of Punjabi EnConverter is performed by using test collections, *i.e.*, a set of one thousand Punjabi sentences are created with their corresponding set of expected UNL expressions as output. For this purpose Spanish UNL Language Server (Spanish Language Center, 2004) and agricultural domain threads developed by IIT Bombay, India are considered as gold-standard EnConverters.

Spanish Language Server contains English sentences with their corresponding UNL expressions generated by the system. These English sentences were translated manually into equivalent Punjabi sentences and then inputted to the proposed Punjabi EnConverter system for their EnConversion to UNL. The UNL expression generated by proposed system is compared with the UNL expression generated by Spanish Language Server. The agricultural domain threads developed by IIT Bombay have Hindi language sentences with their equivalent UNL expressions. These Hindi sentences were manually translated into Punjabi language and then processed in Punjabi EnConverter for their equivalent UNL expressions. Similarly, the comparison has been performed for the UNL expressions generated by proposed system with the corresponding UNL expressions given at agricultural domain threads.

The two UNL expressions match with each other if the UNL relations, including associated UWs and UNL attributes present in the expressions are same (Jain and Damani, 2009). It has been seen that proposed system handles the resolution of UNL relations and generation of attributes for these sentences with a very reasonable accuracy. Results of these evaluations on the basis of each UNL relation are depicted graphically in Figure 6.1.

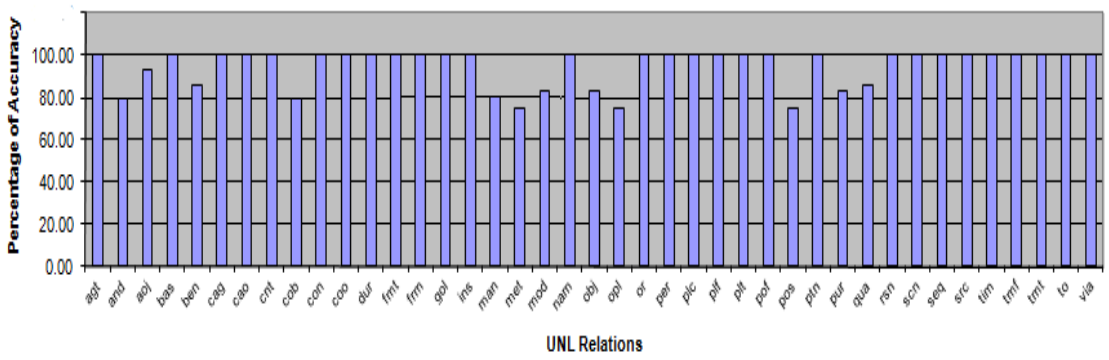

Figure 6.1: Results of evaluation of Punjabi EnConveter for UNL relations

6.4 Evaluation of Punjabi DeConverter

Proposed Punjabi DeConverter is evaluated by inputting UNL expressions generated by Punjabi EnConverter to it and the output of Punjabi DeConverter is compared with input Punjabi sentence given to Punjabi EnConverter. The approach followed in this work for evaluating UNL based MT system for Punjabi language is given in Figure 6.2. The two Punjabi sentences (original sentence inputted to Punjabi EnConverter and the sentence generated by DeConverter) are compared and the system is evaluated based on different tests on these outputs.

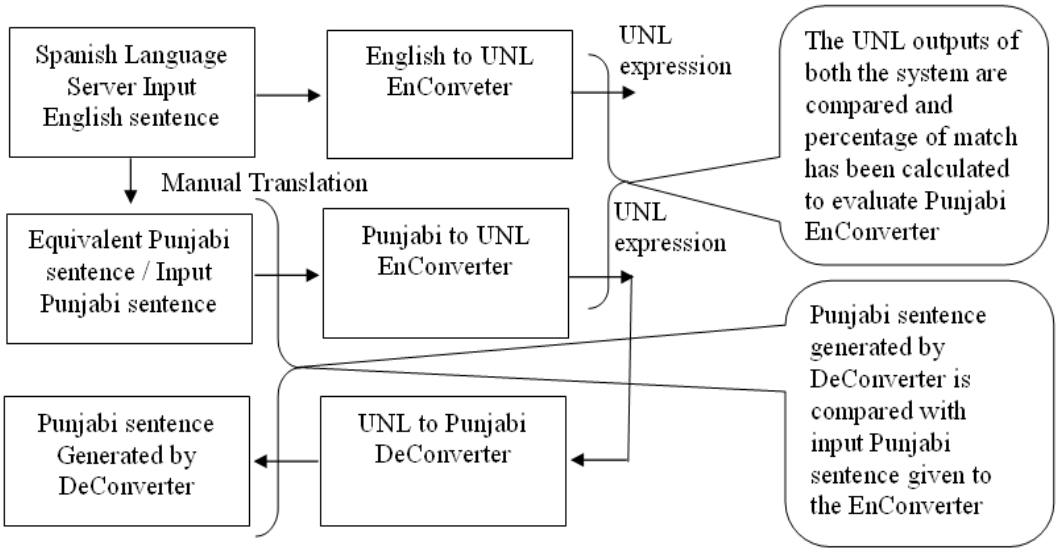

Figure 6.2: Approach for evaluating UNL based MT system for Punjabi language

6.5 Tests for the evaluation

There are number of tests available for evaluating MT systems. Slype (1979) describes subjective tests and quantitative tests for evaluation of an MT system. Subjective tests

like adequacy and fluency tests have been performed on the proposed system. BLEU score has been calculated to evaluate the quality of output. The quantitative test, namely, error analysis has also been performed by calculating Sentence Error Rate (SER) and Word Error Rate (WER) of the proposed UNL based MT system for Punjabi language.

6.5.1 Fluency test

Fluency refers to the degree to which the target is well formed according to the rules of target language grammar. A fluent segment is one that is well-formed grammatically, contains correct spellings, adheres to common use of terms, titles and names, is intuitively acceptable and can be sensibly interpreted by a native speaker of that language (LDC, 2005). There are number of discussions available in the literature regarding the choice for point of scale in the evaluation. It has been suggested that one can use 5-point scale for these tests (LDC, 2005). Singh *et al.* (2007) and Goyal (2010), however, suggested that 5-point scale is a too fine-grained scale and in this scale the distinction may result in evaluators worrying a lot about making an accurate call, and intuitive judgment may get affected. It also makes the evaluation even more subjective. Thus, a 4-point scale has been followed for evaluation of the proposed system as followed by Singh *et al.* (2007) and Goyal (2010). The details of 4-point scale for the fluency score is given in Table 6.1.

Table 6.1: 4-point scale of fluency score

Score	Level	Description
4	Perfect	Good grammar
3	Fair	Easy-to-understand but flawed grammar
2	Acceptable	Broken, understandable with effort
1	Nonsense	Incomplete

6.5.2 Adequacy test

Adequacy refers to the degree to which the information present in original sentence is also communicated in translated sentence. Thus, for adequacy judgments, the gold-standard will serve as a proxy for the original source language text. For the evaluation of proposed Punjabi DeConverter, the sentence generated by the system is compared with the input Punjabi sentence of EnConverter, whose UNL expression is fed to the Punjabi

DeConverter for DeConversion. Thus, for the adequacy test the evaluator is also presented with the gold-standard translation, *i.e.*, the original input Punjabi sentence to Punjabi EnConverter, with reference to which they can compare the generated output. Again, a 4-point scale has been followed for the adequacy test of the proposed system as followed by Singh *et al.* (2007) and Goyal (2010). The details of 4-point scale for the adequacy score is given in Table 6.2.

Table 6.2: 4-point scale of adequacy score

Scale	Level	Description
4	All	No loss of meaning
3	Most	Most of the meaning is conveyed
2	Some	Some of the meaning is conveyed
1	None	Hardly any meaning is conveyed

6.5.3 BLEU score

Bilingual Evaluation Understudy (BLEU) score is commonly used for automatically evaluating the quality of output produced by an MT system. BLEU is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Scores are calculated for individual translated segments, generally sentences, by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to estimate the translation's overall quality. In BLEU score intelligibility or grammatical correctness are not taken into account. BLEU's output is always a number between 0 and 1. This value indicates how similar the original and translated sentences are. In case BLEU score is closer to one, the two sentences are closer to each other.

During the processing of BLEU score the *n*-grams of the translated sentence is compared with the *n*-grams of the original sentences and system counts the number of matches between both the original and translated sentences. These matches are position independent. If we have higher matches, the quality of translation system is better. The metric calculates scores for individual sentences, and then averages these scores over the whole corpus in order to reach a final score (Goyal, 2010).

6.5.4 Tests based on quantitative metrics

Quantitative metrics are based on the analysis of errors made by the MT system. It performs error analysis to establish how seriously errors affect the translation output. The error analysis includes Word Error Rate (WER) and Sentence Error Rate (SER). Word Error Rate (WER) is defined as percentage of words which are to be inserted, deleted, or replaced in the translation in order to obtain the original sentence. Sentence Error Rate (SER) is defined as percentage of sentences, whose translations have not matched in an exact manner with those of reference (Goyal, 2010).

6.6 Experiments

It is also very important to choose appropriate evaluators for performing the tests on the proposed system. The proposed system has been evaluated by ten evaluators having a good knowledge of Punjabi and a basic knowledge of UNL system. As suggested in LDC (2005) and by Singh *et al.* (2007), the evaluators were asked to provide their intuitive reaction to the output and to work as quickly as comfortable. First of all, fluency judgments have been taken. The evaluators did not have any clue about the original source sentence for these judgments. After fluency judgments, the evaluators were asked to look at the original source sentences for adequacy judgments.

6.7 Results for fluency score

The fluency score of proposed system is 3.61 (on a 4-point scale). The response by evaluators is further analyzed and following are some important findings for the fluency test.

- 73.50% sentences got a score 4, *i.e.*, these are perfect having no grammatical mistake.
- 15.50% sentences got a score 3, *i.e.*, these are fair and easy to understand.
- 9.50% sentences got a score 2, *i.e.*, these are acceptable and are understandable with effort.
- 1.50% sentences got a score 1, *i.e.*, these are hard to understand.

Here, it is worth mentioning that proposed system generated 89.0% grammatically correct sentences. These sentences are those that have a score of 3 or above.

6.8 Results for adequacy score

The adequacy score of proposed system is 3.70 (on a 4-point scale). The response by the evaluators has again been further analyzed and following are some important findings for the adequacy test.

- 79.2% sentences got a score 4, *i.e.*, there is no loss of meaning during their translation.
- 12.8% sentences got a score 3, *i.e.*, most of the meaning of these source sentences is conveyed in the translated sentence.
- 7.0% sentences got a score 2, *i.e.*, some of the meaning of these source sentences is conveyed in the translated sentence.
- 1.0% sentences got a score 1, *i.e.*, hardly any meaning of these source sentences is conveyed in the translated sentence.

Here, it is worth mentioning that proposed system generated 92% sentences that are faithful to the original sentences. These sentences are those that have a score of 3 or above.

We have also observed that there is a strong correlation between fluency and adequacy scores. This relation between adequacy and fluency is explored further in Figure 6.3. It shows the distribution of adequacy scores for various values of fluency.

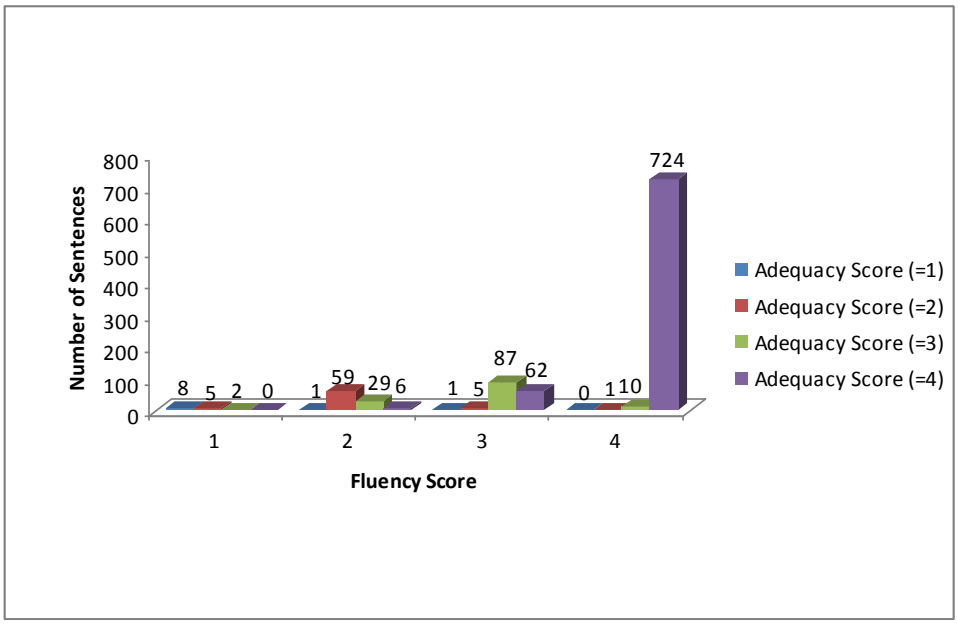

Figure 6.3: Distribution of adequacy scores for various fluency scores

6.9 Results for BLEU score

The quality of proposed system is also evaluated on the basis of BLEU score. The original sentences inputted to Punjabi EnConverter have been considered as reference sentences and corresponding sentences generated by Punjabi DeConverter have been considered as machine generated sentences. The proposed machine translation system is able to achieve a BLEU score of 0.72.

6.10 Error Analysis

The error analysis has also been performed for the proposed system after calculating fluency, adequacy and BLEU score. The error analysis has been performed for classified error list that includes the errors like addition of extra words, removal of words, wrong order of words and wrong choice of words in the generated output. All these errors in generated output have been identified and noted. After robust analysis, word error rate is found to be 5.43%. The results of word error rate are depicted in the graph given in Figure 6.4.

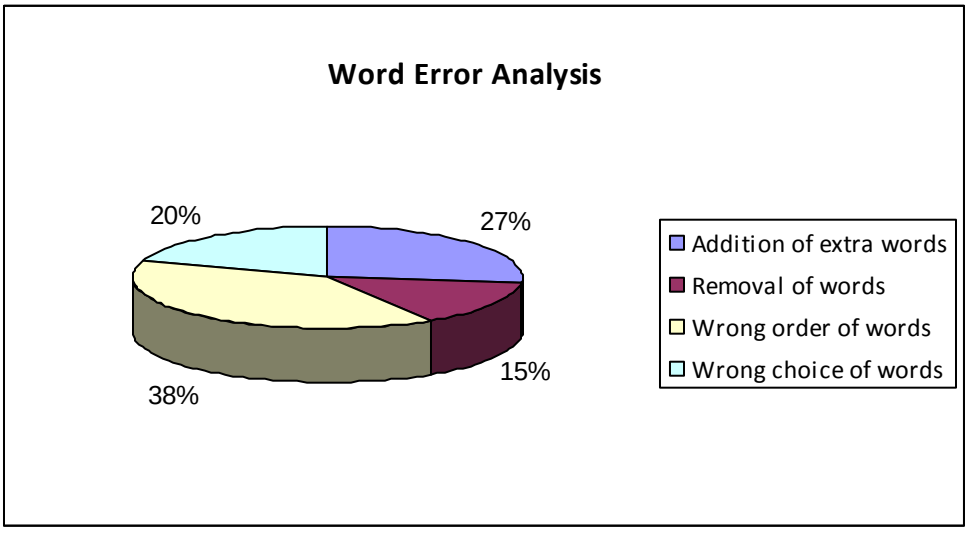

Figure 6.4: Word Error Analysis of the proposed system

From the above figure, it can be concluded that majority of the errors are due to wrong word order. The order of words in the generated sentence is decided by the syntax planning phase of Punjabi DeConverter. This order is governed by a matrix based on priority of UNL relations sharing a common parent. Since, Punjabi is a free order language the change in the word order will not severely affect the accuracy of the system. The function word insertion rule base of Punjabi DeConverter is responsible for insertion

of extra words in the generated output. The morphology generation rules are responsible for removal of words and wrong choice of words in the generated output.

After word error rate analysis, the sentence error rate analysis has been performed by calculating the number of sentences, whose translations have not matched in an exact manner with those of reference sentences. The proposed system has a sentence error rate of 20.8%.

However, in case of Punjabi EnConverter the wrong parsing of input sentence by Punjabi shallow parser results into generation of incorrect UNL expression, which causes errors in the generated output. Sometimes, the non availability of semantic information of verbs in lexicon also causes incorrect firing of rules, which result into incorrect output.

Chapter Summary

The proposed system has been evaluated on the basis of subjective tests and quantitative metrics. From the analysis in this chapter, it has been concluded that proposed system generates 89.0% grammatically correct sentences; and generates 92.0% sentences that are faithful to the original sentences. The system could achieve a fluency score of 3.61 (on a 4-point scale) and adequacy score of 3.70 (on a 4-point scale). The proposed system is able to achieve a BLEU score of 0.72. The proposed system has a word error rate of 5.43% and sentence error rate of 20.8%. These scores of the proposed system can be improved further by improving the rule base and lexicon.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

This thesis explains the development of a UNL based MT system for Punjabi language. UNL based MT systems follow the interlingua approach of MT. The approach in UNL revolves around the development of an EnConverter and a DeConverter for a natural language. The EnConverter is used to convert a given sentence in natural language to an equivalent UNL expression; and the DeConverter is used to convert a given UNL expression to an equivalent natural language sentence. UNL system has the potential to bridge the language barriers across the world with the development of 2*n* components, while traditional approaches require the *n**(*n*-1) components, where *n* is the number of languages. UNL represents the information sentence by sentence in the form of Universal Words (UWs), UNL relations and UNL attributes. The concepts are expressed by UWs and UNL relations are used to specify the role of each word in a sentence. The subjective meanings intended by the author are expressed through UNL attributes. UNL system makes use of word dictionaries in the form of Language-UW lexicon of respective languages for its processing. A major outcome of this research work is the development of Punjabi EnConverter, Punjabi DeConverter and a web interface for online EnConversion and DeConversion process. 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 developed in this work can convert simple and complex Punjabi sentences to equivalent UNL expressions.

Punjabi DeConverter involves UNL parser, lexeme selection, morphology generation,

function word insertion and syntax planning phases. First stage of a DeConverter is UNL parser which parses the input UNL expression to build a node-net from the input UNL expression. During lexeme selection phase, Punjabi root words and their dictionary attributes are selected for the given UWs in the input UNL expression from the Punjabi-UW dictionary. After that, the nodes are ready for generation of morphology according to the target language in the morphology phase. The proposed system makes use of morphology rule base for Punjabi language to handle attribute label resolution morphology; relation label resolution morphology; and noun, adjective, pronoun and verb morphology. In function word insertion phase, the function words are inserted to the morphed words. Finally, the syntax planning phase is used to define the word order in the generated sentence so that output matches with a natural language sentence. Algorithms and pseudocodes have been implemented for syntax planning for simple sentences, syntax planning of UNL graph with a scope node, untraversed parent, multiple parents, special cases in syntax planning and clausal sentences. All these algorithms and pseudocodes used in different phases of Punjabi EnConverter and DeConverter have been implemented in Java to develop the proposed system.

The proposed system has been tested for one thousand sentences. These sentences have been considered in such a way that generation of all possible UNL relations and attributes can be tested. These sentences include the sentences taken from Spanish UNL Language Server and also from agricultural domain threads developed by IIT Bombay, India. The proposed system has achieved a fluency score of 3.61 (on a 4-point scale) and an adequacy score of 3.70 (on a 4-point scale). The proposed system is able to achieve a BLEU score of 0.72. The proposed system has a word error rate of 5.43% and sentence error rate of 20.8%.

7.2 Limitations of the proposed system

In this section, some of the limitations of the developed MT system are presented.

- The accuracy of Punjabi EnConverter is highly dependent on the accuracy of Punjabi shallow parser used for the parsing of input Punjabi sentence. If it does not parse the input sentence correctly, then the MT system will also not be able to generate its equivalent UNL expression correctly.

- Firing of an accurate EnConversion and DeConversion rule sometime requires semantic information of verbs. The non availability of this information for some verbs in lexicon results into incorrect resolution of UNL relation between UWs.
- The system is not able to resolve ambiguous words; it altogether depends upon Punjabi shallow parser for resolving their part of speech information.
- The system is not able to handle very large and complex sentences.
- The system is not able to perform automatic EnConversion and DeConversion of web pages.

7.3 Future scope of the proposed work

In future, this work can further be enriched by incorporating following components into the proposed system. Some of the work that can be carried out in future includes following.

- In case of incorrect parsing of input sentence by Punjabi shallow parser, an interactive system can be designed to override the part of speech information given by the parser.
- The coverage and accuracy of system can further be improved by expanding the Punjabi-UW dictionary and enriching it with more semantic information.
- The rule base can further be improved linguistically to increase the accuracy of the system and to handle very large and complex sentences.
- The ambiguous words can be handled by adding a word sense disambiguation module to the proposed Punjabi EnConverter.
- A UNL based Punjabi Language Server can be developed that may host the EnConverter and DeConverter of Punjabi language. It will start conversions after receiving requests from any web application, and provide the results when the conversions are completed.
- A UNL Proxy Server can be developed to serve as a filter to allow Internet browsers to recognize web pages written in UNL and then selecting an appropriate Language Server on the Internet so that the document can be read in a natural language.
- The proposed system can further be tested on other languages by using corresponding language's rule base and lexicon.

Chapter Summary

This chapter concludes the work done during this thesis. This thesis explains the development of a Punjabi-UNL EnConverter and UNL-Punjabi DeConverter. All the algorithms and pseudocodes used in different phases of Punjabi EnConverter and DeConverter have been implemented in Java to develop the proposed system. A web interface for online EnConversion and online DeConversion tasks has also been developed. The proposed system has been tested for one thousand sentences. The proposed system has achieved a fluency score of 3.61 (on a 4-point scale) and an adequacy score of 3.70 (on a 4-point scale). The proposed system has a BLEU score of 0.72, a word error rate of 5.43% and sentence error rate of 20.8%.

There are some limitations of the proposed system that include dependence of Punjabi EnConverter on the accuracy of Punjabi shallow parser, the non availability of semantic information for some verbs in lexicon, dependence on Punjabi shallow parser to resolve ambiguous words and the inability to handle very large and complex sentences.

In future this work can further be enriched by incorporating an interactive system to override incorrect information given by Punjabi shallow parser in case of incorrect parsing of input sentence. Punjabi-UW dictionary can further be enhanced with addition of more semantic information. The proposed system can further be improved by adding a word sense disambiguation module to resolve ambiguities. Setting up a UNL based Punjabi Language Server for automatic EnConversion and DeConversion of web pages can further increase its usage.

Appendix-A

Some important EnConversion rules

S.No	EnConversion rules
1.	+{ADJ:+VAUX,+.@present.@sg:null}{[ਹੈ]:null:null}
2.	+{V,f,custom:+.@past.@custom.@sg.@female:null}{[ਸੀ]:null:null}
3.	+{V,f,custom:+.@future.@custom.@sg.@female:null}{[ਹੋ ਏਗਾ]:null:null}
4.	+{V:+.@future.@sg.@male:null}{[ਹੋ ਏਗਾ]:null:null}
5.	+{V:+.@past.@sg:null}{[ਸੀ]:null:null}
6.	>{N,TIME,TOON:null:tmf}{V:+TMFRES:null}
7.	>{N,TIME,TOON:null:tmf}{N,ST:+TMFRES:null}
8.	>{N,DUR,TOON:null:tmf}{V:+TMFRES:null}
9.	>{N,DUR,TOON:null:tmf}{V:+TMFRES:null}
10.	>{N,DUR,TOON:null:tmf}{N,KARMSG:+TMFRES:null}
11.	>{PRON,MOD:null:tmf}{N,TIME:+TMFRES:null}
12.	>{NUM,^TIME,^DUR:null:qua}{N,^PLC:+QUARES:null}
13.	>{N,^PLC,^ANIMT:null:qua}{NUM,^TIME,^DUR:+QUARES:null}
14.	>{ADJ:null:qua}{N,INANI,UNIT:+QUARES:null}
15.	>{ADJ,QUA:null:qua}{N,INANI:+QUARES:null}
16.	>{N,PLC,FRM:null:frm}{N:+FRMRES:null}
17.	>{N,POS:null:pof}{N,POF:+POFRES:null}
18.	>{N,CAG,COMMUNITY:null:ptn}{V:+PTNRES:null}
19.	>{N,CAG:null:ptn}{ADJ,KAR:+PTNRES:null}
20.	>{N,CAG:null:ptn}{V,NOACT:+PTNRES:null}
21.	>{PRON,POS,sg,f:+.@sg.@female:pos}{N,^POF,^NABS:+POSRES:null}
22.	>{PRON,POS,pl,m:+.@pl.@male:pos}{N,^POF,^NABS:+POSRES:null}
23.	>{PRON,POS,f:+.@female:pos}{N,^POF,^NABS:+POSRES:null}
24.	>{N,POS,^RSN,^CAG,^PLC:null:pos}{N,^POF,^NABS:+POSRES:null}
25.	<{N,ACT,INANI:null:null}{N,RSN,HO:+RSNRES:rsn}
26.	>{N,RSN:null:rsn}{N,HO:+RSNRES:null}

27.	>{N,RSN:null:rsn}{N,VAUX:+RSNRES:null}
28.	>{N,RSN:null:rsn}{V:+RSNRES:null}
29.	>{PRON,RSN:null:rsn}{V:+RSNRES:null}
30.	>{N,RSN:null:rsn}{N,KAR:+RSNRES:null}
31.	>{PRON,RSN:null:rsn}{N,KAR:+RSNRES:null}
32.	>{PRON,^ANIMT,^CASE:null:mod}{N,^KAR,^TIME:+MODRES:null}:
33.	>{PRON,POS:null:mod}{N,NABS,^KAR,^TIME:+MODRES:null}:
34.	>{ADJ:null:mod}{N,ABS:+MODRES:null}
35.	>{N,PROPER,^PLC:null:nam}{N,^PLC:+NAMRES:null}
36.	>{N,PROPER,TCL:null:nam}{N:+NAMRES,+TCL:null}
37.	<{N:+KAR:null}{v,KAR:null:null}
38.	+{N,KAR:+.@past.@sg.@male:null}{[ਇਆ]:null:null}
39.	+{N,KARMSG:+.@past.@sg.@male:null}{[ਇਆ]:null:null}
40.	+{N,KARMPL:+.@past.@pl.@male:null}{[ਇਆ]:null:null}
41.	+{N,KARFSG:+.@past.@sg.@female:null}{[ਇਆ]:null:null}
42.	+{N,KARFPL,^pl:+.@past.@pl.@female:null}{[ਇਆ]:null:null}
43.	+{N,KARFPL,pl:+.@past.@female:null}{[ਇਆ]:null:null}
44.	+{ADJ,KARMSG:+.@past.@sg.@male:null}{[ਇਆ]:null:null}
45.	+{ADJ,KARMPL:+.@past.@pl.@male:null}{[ਇਆ]:null:null}
46.	+{ADJ,KARFSG:+.@past.@sg.@female:null}{[ਇਆ]:null:null}
47.	+{ADJ,KARFPL:+.@past.@pl.@female:null}{[ਇਆ]:null:null}
48.	+{ADJ,KAR:+.@past.@sg.@male:null}{[ਇਆ]:null:null}
49.	+{V:+.@future.@sg.@male:null}{[ੇਗਾ]:null:null}
50.	+{V:+.@future.@sg.@female:null}{[ੇਗੀ]:null:null}
51.	+{V:+.@past.@custom.@sg.@female:null}{[ਿਚੱਤੀ]:null:null}
52.	+{V:+.@past.@custom.@pl.@female:null}{[ਿਚੱਤੀਆਂ]:null:null}
53.	+{V:+.@past.@custom.@pl.@male:null}{[ਿਚੱਤੇ]:null:null}

54.	>{N,INANI,ACT,TCL:null:agt}{V,^VOCURR,^COO:+AGTRES,+TCL:null}
55.	>{N,INANI,ACT:null:agt}{N,KAR,^VOCURR,^COO:+AGTRES:null}
56.	>{N,INANI,MACH:null:agt}{V,^VOCURR,^COO:+AGTRES:null}
57.	>{N,INANI,MACH:null:agt}{N,KARFSG,^VOCURR,^COO:+AGTRES:null}
58.	>{N,INANI,MACH:null:agt}{N,KARFPL,^VOCURR,^COO:+AGTRES:null}
59.	>{N,INANI,ACT,TCL:null:agt}{V,^VOCURR,^COO:+AGTRES,+TCL:null}
60.	>{N,INANI,ACT:null:agt}{N,KAR,^VOCURR,^COO:+AGTRES:null}
61.	>{N,CASE,^CAG:null:agt}{V,^AGTRES,^VOCURR,^COO:+AGTRES:null}
62.	>{PRON,BEN:null:ben}{V:+BENRES:null}
63.	>{N,BEN:null:ben}{V:+BENRES:null}
64.	>{PRON,BEN,TCL:null:ben}{V:+BENRES,+TCL:null}
65.	>{PRON,ANIMT,NOO:null:ben}{V,^COB,BEN:+BENRES:null}
66.	>{N,ANIMT,NOO:null:ben}{V,^COB,BEN:+BENRES:null}
67.	+{N:+CASE:null}{case:+null:null}
68.	+{N:+.@pl.@female:null}{[ਆਂ]:null:null}
69.	+{KAR:null:null}{[ਣ]:null:null}
70.	+{PRON:+CASE:null}{case:+null:null}
71.	+{V:+.@present.@sg.@male:null}{[ਂਦਾ ਹੈ]:null:null}
72.	+{V:+.@present.@sg.@male:null}{[ਂਧਾ ਹੈ]:null:null}
73.	+{V:+.@custom.@present.@sg.@female:null}{[ਦੀ ਹੈ]:null:null}
74.	+{V:+.@future.@sg.@male:null}{[ਵਾਂਗਾ]:null:null}
75.	+{V,m,sg:+.@future.@sg.@male:null}{[ਏਗਾ]:null:null}
76.	+{V:+.@future:null}{[ਕਰੇਗੀ]:null:null}
77.	-{TCL,^V,^N,^PRON,^ADJ,^ADV:null:null}{PRON:+TCL:null}
78.	-{TCLB,^V,^N,^PRON,^ADJ,^ADV:null:null}{PRON:+TCLB:null}
79.	-{MCLB,^V:null:null}{PRON,^MCLB:+MCLB:null}
80.	-{MCL:null:null}{PRON,^MCL:+MCL:null}
81.	-{PCLB,^V:null:null}{V,^PCLB:+PCLB:null}
82.	-{PCL:null:null}{PRON,^PCL:+PCL:null}

83.	-{ADJCLB,^V:null:null}{V,^ADJCLB:+ADJCLB:null}
84.	-{CCLB,^V,^N,^PRON,^ADJ,^ADV:null:null}{PRON:+CCLB:null}
85.	-{CCLB,^V,^N,^PRON,^ADJ,^ADV:null:null}{N:+CCLB:null}
86.	-{CCL,^V,^N,^PRON,^ADJ,^ADV:null:null}{PRON:+CCL:null}
87.	-{CCL,^V,^N,^PRON,^ADJ,^ADV:null:null}{N:+CCL:null}
88.	-{ADCL,CCL:null:null}{PRON:+CCL:null}
89.	>{V,TCLB:+.@entry:null}{V,TCL:null:tim}
90.	<{V,MCLB:null:null}{V,MCL:+.@entry:man}
91.	<{V,PCLB:+.@entry:null}{V,PCL:+.@entry:plc}
92.	<{V,ADJCLB:+PRONADD:null}{PRON:null:null}
93.	<{V,ADJCLB,PRONADD:null:aoj}{ADJ:+.@entry:null}
94.	>{ADJ,RESTRICT:null:mod}{N,INANI:+MODRES:null}
95.	>{ADJ,MOD:null:mod}{N,INANI:+MODRES:null}
96.	>{ADJ,BAS:null:mod}{N,INANI:+MODRES:null}
97.	>{N:null:mod}{N,RESTRICT:+MODRES:null}
98.	>{PRON,ANIMT,FIRST,dis:null:mod}{N,INANI:+MODRES:null}
99.	>{N,KAR,MODRES:null:mod}{N:+MODRES:null}
100.	>{N:null:mod}{N,INANI,ABS-ACT:+MODRES:null}
101.	>{N,NPOS:null:mod}{N,INANI:+MODRES:null}
102.	>{N,PSP:null:mod}{N,INANI,ST:+MODRES:null}
103.	>{N,PSP:null:mod}{N,INANI,MOD:+MODRES:null}
104.	>{N,^PLC:null:mod}{N,INANI,ST,MOD:+MODRES:null}
105.	>{N,TOON,^FMTRES:null:src}{V:+SRCRES:null}
106.	>{ADJ,TOON,^FMTRES:null:src}{V:+SRCRES:null}
107.	>{ADJ,TOON,^FMTRES:null:src}{V:+SRCRES:null}
108.	>{N,TOON,^FMTRES:null:src}{N,KAR:+SRCRES:null}
109.	>{ADJ,TOON,^FMTRES:null:src}{N,KAR:+SRCRES:null}
110.	>{N,WICH,^PLC,^TIME:null:gol}{V,FST:+GOLRES:null}
111.	>{N,SRCRES,^PLC,^TIME:null:gol}{V,FST:+GOLRES:null}
112.	>{ADJ,WICH,^PLC,^TIME:null:gol}{V,FST:+GOLRES:null}

Appendix-B

Attribute label morphology for masculine gender

Morphology	Per-son	Nu-mber	UNL Attributes	Example sentence
ਿਆ ਸੀ *iā sī*	1	SG	@past, @custom,1per,@sg	ਮੈਂ ਖੇਡਿਆ ਸੀ । *maiṃ khēḍiā sī.*
	2	SG	@past, @custom,2per,@sg	ਤੂੰ ਖੇਡਿਆ ਸੀ । *tūṃ khēḍiā sī.*
	3	SG	@past, @custom,3per,@sg	ਮੁੰਡਾ ਖੇਡਿਆ ਸੀ । *muṇḍā khēḍiā sī.*
ਏ ਸੀ *ē sī,* ਏ ਸਨ *ē san*	1	PL	@past, @custom,1per,@pl	ਅਸੀਂ ਖੇਡੇ ਸੀ । *asīṃ khēḍē sī.*
	2	PL	@past, @custom,2per,@pl	ਤੁਸੀਂ ਖੇਡੇ ਸੀ । *tusīṃ khēḍē sī.*
	3	PL	@past, @custom,3per,@pl	ਮੁੰਡੇ ਖੇਡੇ ਸਨ । *muṇḍē khēḍē san.*
ਰਿਹਾ ਸੀ *rihā sī*	1	SG	@past, @continous,1per,@sg	ਮੈਂ ਖੇਡ ਰਿਹਾ ਸੀ । *maiṃ khēḍ rihā sī.*
	2	SG	@past, @continous,2per,@sg	ਤੂੰ ਖੇਡ ਰਿਹਾ ਸੀ । *tūṃ khēḍ rihā sī.*
	3	SG	@past, @continous,3per,@sg	ਮੁੰਡਾ ਖੇਡ ਰਿਹਾ ਸੀ । *muṇḍā khēḍ rihā sī.*
ਰਹੇ ਸੀ *rahē sī,* ਰਹੇ ਸਨ *rahē san*	1	PL	@past, @continous,1per,@pl	ਅਸੀਂ ਖੇਡ ਰਹੇ ਸੀ । *asīṃ khēḍ rahē sī.*
	2	PL	@past, @continous,2per,@pl	ਤੁਸੀਂ ਖੇਡ ਰਹੇ ਸੀ । *tusīṃ khēḍ rahē sī.*
	3	PL	@past, @continous,3per,@pl	ਮੁੰਡੇ ਖੇਡ ਰਹੇ ਸਨ । *muṇḍē khēḍ rahē san.*
ਚੁੱਕਾ ਸੀ *cukkā sī*	1	SG	@past, @complete,1per,@sg	ਮੈਂ ਖੇਡ ਚੁੱਕਾ ਸੀ । *maiṃ khēḍ cukkā sī.*
	2	SG	@past, @complete,2per,@sg	ਤੂੰ ਖੇਡ ਚੁੱਕਾ ਸੀ । *tūṃ khēḍ cukkā sī.*
	3	SG	@past, @complete,3per,@sg	ਮੁੰਡਾ ਖੇਡ ਚੁੱਕਾ ਸੀ । *muṇḍā khēḍ cukkā sī.*

ਚੁੱਕੇ ਸੀ *cukkē sī*, ਚੁੱਕੇ ਸਨ *cukkē san*	1	PL	@past, @complete,1per,@pl	ਅਸੀਂ ਖੇਡ ਚੁੱਕੇ ਸੀ । *asīṃ khēḍ cukkē sī.*
	2	PL	@past, @complete,2per,@pl	ਤੁਸੀਂ ਖੇਡ ਚੁੱਕੇ ਸੀ । *tusīṃ khēḍ cukkē sī.*
	3	PL	@past, @complete,3per,@pl	ਮੁੰਡੇ ਖੇਡ ਚੁੱਕੇ ਸਨ । *muṇḍē khēḍ cukkē san.*
ਦਾ ਰਹਿੰਦਾ ਸੀ *dā rahindā sī*	1	SG	@past, @repeat,1per,@sg	ਮੈਂ ਖੇਡਦਾ ਰਹਿੰਦਾ ਸੀ । *maiṃ khēḍdā rahindā sī.*
	2	SG	@past, @repeat,2per,@sg	ਤੂੰ ਖੇਡਦਾ ਰਹਿੰਦਾ ਸੀ । *tūṃ khēḍdā rahindā sī.*
	3	SG	@past, @repeat,3per,@sg	ਮੁੰਡਾ ਖੇਡਦਾ ਰਹਿੰਦਾ ਸੀ । *muṇḍā khēḍdā rahindā sī.*
ਦੇ ਰਹਿੰਦੇ ਸੀ *dē rahindē sī*, ਦੇ ਰਹਿੰਦੇ ਸਨ *dē rahindē san*	1	PL	@past, @repeat,1per,@pl	ਅਸੀਂ ਖੇਡਦੇ ਰਹਿੰਦੇ ਸੀ । *asīṃ khēḍdē rahindē sī.*
	2	PL	@past, @repeat,2per,@pl	ਤੁਸੀਂ ਖੇਡਦੇ ਰਹਿੰਦੇ ਸੀ । *tusīṃ khēḍdē rahindē sī.*
	3	PL	@past, @repeat,3per,@pl	ਮੁੰਡੇ ਖੇਡਦੇ ਰਹਿੰਦੇ ਸਨ । *muṇḍē khēḍdē rahindē san.*
ਦਾ ਹਾਂ *dā hāṃ*, ਦਾ ਹੈਂ *dā haiṃ*, ਦਾ ਹੈ *dā hai*	1	SG	@present, @custom,1per,@sg	ਮੈਂ ਖੇਡਦਾ ਹਾਂ । *maiṃ khēḍdā hāṃ.*
	2	SG	@present, @custom,2per,@sg	ਤੂੰ ਖੇਡਦਾ ਹੈਂ । *tūṃ khēḍdā haiṃ.*
	3	SG	@present, @custom,3per,@sg	ਮੁੰਡਾ ਖੇਡਦਾ ਹੈ । *muṇḍā khēḍdā hai.*
ਦੇ ਹਾਂ *dē hāṃ*, ਦੇ ਹੋ *dē hō*, ਦੇ ਹਨ *dē han*	1	PL	@present, @custom,1per,@pl	ਅਸੀਂ ਖੇਡਦੇ ਹਾਂ । *asīṃ khēḍdē hāṃ.*
	2	PL	@present, @custom,2per,@pl	ਤੁਸੀਂ ਖੇਡਦੇ ਹੋ । *tusīṃ khēḍdē hō.*
	3	PL	@present, @custom,3per,@pl	ਮੁੰਡੇ ਖੇਡਦੇ ਹਨ । *muṇḍē khēḍdē han.*
ਰਿਹਾ ਹਾਂ *rihā hāṃ*,	1	SG	@present,@continous,1per,@sg	ਮੈਂ ਖੇਡ ਰਿਹਾ ਹਾਂ । *maiṃ khēḍ rihā hāṃ.*

ਰਿਹਾ ਹੈਂ *rihā haiṃ*, ਰਿਹਾ ਹੈ *rihā hai*	2	SG	@present,@continous,2per,@sg	ਤੂੰ ਖੇਡ ਰਿਹਾ ਹੈਂ । *tūṃ khēḍ rihā haiṃ.*
	3	SG	@present,@continous,3per,@sg	ਮੁੰਡਾ ਖੇਡ ਰਿਹਾ ਹੈ । *muṇḍā khēḍ rihā hai.*
ਰਹੇ ਹਾਂ *rahē hāṃ*, ਰਹੇ ਹੋ *rahē hō*, ਰਹੇ ਹਨ *rahē han*	1	PL	@present,@continous,1per,@pl	ਅਸੀਂ ਖੇਡ ਰਹੇ ਹਾਂ । *asīṃ khēḍ rahē hāṃ.*
	2	PL	@present,@continous,2per,@pl	ਤੁਸੀਂ ਖੇਡ ਰਹੇ ਹੋ । *tusīṃ khēḍ rahē hō.*
	3	PL	@present,@continous,3per,@pl	ਮੁੰਡੇ ਖੇਡ ਰਹੇ ਹਨ । *muṇḍē khēḍ rahē han.*
ਚੁੱਕਾ ਹਾਂ *cukkā hāṃ*, ਚੁੱਕਾ ਹੈਂ *cukkā haiṃ*, ਚੁੱਕਾ ਹੈ *cukkā hai*	1	SG	@present,@complete,1per,@sg	ਮੈਂ ਖੇਡ ਚੁੱਕਾ ਹਾਂ । *maiṃ khēḍ cukkā hāṃ.*
	2	SG	@present,@complete,2per,@sg	ਤੂੰ ਖੇਡ ਚੁੱਕਾ ਹੈਂ । *tūṃ khēḍ cukkā haiṃ.*
	3	SG	@present,@complete,3per,@sg	ਮੁੰਡਾ ਖੇਡ ਚੁੱਕਾ ਹੈ । *muṇḍā khēḍ cukkā hai.*
ਚੁੱਕੇ ਹਾਂ *cukkē hāṃ*, ਚੁੱਕੇ ਹੋ *cukkē hō*, ਚੁੱਕੇ ਹਨ *cukkē han*	1	PL	@present,@complete,1per,@pl	ਅਸੀਂ ਖੇਡ ਚੁੱਕੇ ਹਾਂ । *asīṃ khēḍ cukkē hāṃ.*
	2	PL	@present,@complete,2per,@pl	ਤੁਸੀਂ ਖੇਡ ਚੁੱਕੇ ਹੋ । *tusīṃ khēḍ cukkē hō.*
	3	PL	@present,@complete,3per,@pl	ਮੁੰਡੇ ਖੇਡ ਚੁੱਕੇ ਹਨ । *muṇḍē khēḍ cukkē han.*
ਦਾ ਰਹਿੰਦਾ ਹਾਂ *dā rahindā hāṃ*, ਦਾ ਰਹਿੰਦਾ ਹੈਂ *dā rahindā haiṃ* ਦਾ ਰਹਿੰਦਾ ਹੈ	1	SG	@present,@repeat,1per,@sg	ਮੈਂ ਖੇਡਦਾ ਰਹਿੰਦਾ ਹਾਂ । *maiṃ khēḍdā rahindā hāṃ.*
	2	SG	@present,@repeat,2per,@sg	ਤੂੰ ਖੇਡਦਾ ਰਹਿੰਦਾ ਹੈਂ । *tūṃ khēḍdā rahindā haiṃ.*
	3	SG	@present,@repeat,3per,@sg	ਮੁੰਡਾ ਖੇਡਦਾ ਰਹਿੰਦਾ ਹੈ । *muṇḍā khēḍdā rahindā hai.*

dā rahindā hai				
ਦੇ ਰਹਿੰਦੇ ਹਾਂ *dē rahindē hāṃ*,	1	PL	@present,@repeat,1per,@pl	ਅਸੀਂ ਖੇਡਦੇ ਰਹਿੰਦੇ ਹਾਂ । *asīṃ khēḍdē rahindē hāṃ.*
ਦੇ ਰਹਿੰਦੇ ਹੋ *dē rahindē hō*,	2	PL	@present,@repeat,2per,@pl	ਤੁਸੀਂ ਖੇਡਦੇ ਰਹਿੰਦੇ ਹੋ । *tusīṃ khēḍdē rahindē hō.*
ਦੇ ਰਹਿੰਦੇ ਹਨ *dē rahindē han*	3	PL	@present,@repeat,3per,@pl	ਮੁੰਡੇ ਖੇਡਦੇ ਰਹਿੰਦੇ ਹਨ । *muṇḍē khēḍdē rahindē han.*
ਦਾ ਹੋਵਾਂਗਾ *dāhōvāṅgā*,	1	SG	@future, @custom,1per,@sg	ਮੈਂ ਖੇਡਦਾ ਹੋਵਾਂਗਾ । *maiṃ khēḍdā hōvāṅgā.*
ਦਾ ਹੋਵੇਂਗਾ *dā hōvēṅgā*,	2	SG	@future, @custom,2per,@sg	ਤੂੰ ਖੇਡਦਾ ਹੋਵੇਂਗਾ । *tūṃ khēḍdā hōvēṅgā.*
ਦਾ ਹੋਵੇਗਾ *dā hōvēgā*	3	SG	@future, @custom,3per,@sg	ਮੁੰਡਾ ਖੇਡਦਾ ਹੋਵੇਗਾ । *muṇḍā khēḍdā hōvēgā.*
ਦੇ ਹੋਵਾਂਗੇ *dē hōvāṅgē*,	1	PL	@future, @custom,1per,@pl	ਅਸੀਂ ਖੇਡਦੇ ਹੋਵਾਂਗੇ । *asīṃ khēḍdē hōvāṅgē.*
ਦੇ ਹੋਵੋਗੇ *dē hōvōgē*,	2	PL	@future, @custom,2per,@pl	ਤੁਸੀਂ ਖੇਡਦੇ ਹੋਵੋਗੇ । *tusīṃ khēḍdē hōvōgē.*
ਦੇ ਹੋਣਗੇ *dē hōṇgē*	3	PL	@future, @custom,3per,@pl	ਮੁੰਡੇ ਖੇਡਦੇ ਹੋਣਗੇ । *muṇḍē khēḍdē hōṇgē.*
ਰਿਹਾ ਹੋਵਾਂਗਾ *rihā hōvāṅgā*,	1	SG	@future,@continous,1per,@sg	ਮੈਂ ਖੇਡ ਰਿਹਾ ਹੋਵਾਂਗਾ । *maiṃ khēḍ rihā hōvāṅgā.*
ਰਿਹਾ ਹੋਵੇਂਗਾ *rihā hōvēṅgā*,	2	SG	@future,@continous,2per,@sg	ਤੂੰ ਖੇਡ ਰਿਹਾ ਹੋਵੇਂਗਾ । *tūṃ khēḍ rihā hōvēṅgā.*
ਰਿਹਾ ਹੋਵੇਗਾ *rahē hōvēgā*	3	SG	@future,@continous,3per,@sg	ਮੁੰਡਾ ਖੇਡ ਰਿਹਾ ਹੋਵੇਗਾ । *muṇḍā khēḍ rihā hōvēgā.*
ਰਹੇ ਹੋਵਾਂਗੇ *rahē*	1	PL	@future,@continous,1per,@pl	ਅਸੀਂ ਖੇਡ ਰਹੇ ਹੋਵਾਂਗੇ । *asīṃ khēḍ rahē hōvāṅgē.*

hōvāṅgē, ਰਹੇ ਹੋਵੋਗੇ *rahē hōvōgē*, ਰਹੇ ਹੋਣਗੇ *rahē hōṇgē*	2	PL	@future,@continous,2per,@pl	ਤੁਸੀਂ ਖੇਡ ਰਹੇ ਹੋਵੋਗੇ । *tusīṃ khēḍ rahē hōvōgē.*
	3	PL	@future,@continous,3per,@pl	ਮੁੰਡੇ ਖੇਡ ਰਹੇ ਹੋਣਗੇ । *muṇḍē khēḍ rahē hōṇgē.*
ਚੁੱਕਿਆ ਹੋਵਾਂਗਾ *cukkiā hōvāṅgā*, ਚੁੱਕਿਆ ਹੋਵੇਂਗਾ *cukkiā hōvēṅgā*, ਚੁੱਕਿਆ ਹੋਵੇਗਾ *cukkiā hōvēgā*	1	SG	@future,@complete,1per,@sg	ਮੈਂ ਖੇਡ ਚੁੱਕਿਆ ਹੋਵਾਂਗਾ । *maiṃ khēḍ cukkiā hōvāṅgā.*
	2	SG	@future,@complete,2per,@sg	ਤੂੰ ਖੇਡ ਚੁੱਕਿਆ ਹੋਵੇਂਗਾ । *tūṃ khēḍ cukkiā hōvēṅgā.*
	3	SG	@future,@complete,3per,@sg	ਉਹ ਖੇਡ ਚੁੱਕਿਆ ਹੋਵੇਗਾ । *uh khēḍ cukkiā hōvēgā.*
ਚੁੱਕੇ ਹੋਵਾਂਗੇ *cukkē hōvāṅgē*, ਚੁੱਕੇ ਹੋਵੋਗੇ *cukkē hōvōgē*, ਚੁੱਕੇ ਹੋਣਗੇ *cukkē hōṇgē*	1	PL	@future,@complete,1per,@pl	ਅਸੀਂ ਖੇਡ ਚੁੱਕੇ ਹੋਵਾਂਗੇ । *asīṃ khēḍ cukkē hōvāṅgē.*
	2	PL	@future,@complete,2per,@pl	ਤੁਸੀਂ ਖੇਡ ਚੁੱਕੇ ਹੋਵੋਗੇ । *tusīṃ khēḍ cukkē hōvōgē.*
	3	PL	@future,@complete,3per,@pl	ਮੁੰਡੇ ਖੇਡ ਚੁੱਕੇ ਹੋਣਗੇ । *muṇḍē khēḍ cukkē hōṇgē.*
ਦਾ ਰਹਿੰਦਾ ਹੋਵਾਂਗਾ *dā rahindā hōvāṅgā*, ਦਾ ਰਹਿੰਦਾ ਹੋਵੇਂਗਾ *dā rahindā*	1	SG	@future,@repeat,1per,@sg	ਮੈਂ ਖੇਡਦਾ ਰਹਿੰਦਾ ਹੋਵਾਂਗਾ । *maiṃ khēḍdā rahindā hōvāṅgā.*
	2	SG	@future,@repeat,2per,@sg	ਤੂੰ ਖੇਡਦਾ ਰਹਿੰਦਾ ਹੋਵੇਂਗਾ । *tūṃ khēḍdā rahindā hōvēṅgā.*
	3	SG	@future,@repeat,3per,@sg	ਮੁੰਡਾ ਖੇਡਦਾ ਰਹਿੰਦਾ ਹੋਵੇਗਾ । *muṇḍā khēḍdā*

hōvēṅgā, ਦਾ ਰਹਿੰਦਾ ਹੋਵੇਗਾ *dā rahindā* *hōvēgā*				*rahindā hōvēgā.*
ਦੇ ਰਹਿੰਦੇ ਹੋਵਾਂਗੇ *dē rahindē hōvāṅgē,* ਦੇ ਰਹਿੰਦੇ ਹੋਵੋਗੇ *dē rahindē hōvōgē,* ਦੇ ਰਹਿੰਦੇ ਹੋਣਗੇ *dē rahindē* *hōṇgē*	1	PL	@future,@repeat,1per,@pl	ਅਸੀਂ ਖੇਡਦੇ ਰਹਿੰਦੇ ਹੋਵਾਂਗੇ । *asīṃ khēḍdē rahindē hōvāṅgē.*
	2	PL	@future,@repeat,2per,@pl	ਤੁਸੀਂ ਖੇਡਦੇ ਰਹਿੰਦੇ ਹੋਵੋਗੇ । *tusīṃ khēḍdē rahindē hōvōgē.*
	3	PL	@future,@repeat,3per,@pl	ਮੁੰਡੇ ਖੇਡਦੇ ਰਹਿੰਦੇ ਹੋਣਗੇ । *muṇḍē khēḍdē rahindē hōṇgē.*

Appendix-C

Some important verb paradigms for first person singular masculine gender

Sr. No	Verb paradigm	Verb Root	Present	Past	Future
1.	ਪੌਣਾ (ਪਾਉਣਾ) *pauṇā* (*pāuṇā*) *'to put'*	ਪਾ *pā*	ਪਾਉਨਾ *pāunā*	ਪਾਇਆ *pāiā*	ਪਾਊਂਗਾ *pāūṅgā*
2.	ਲੌਹਣਾ (ਲਾਹੁਣਾ) *lauhṇā* (*lāhuṇā*) *'to pull down'*	ਲਾਹ *lāh*	ਲਾਹੁਨਾ *lāhunā*	ਲਾਹਿਆ *lāhiā*	ਲਾਹੂੰਗਾ *lāhūṅgā*
3.	ਨ੍ਹੌਣਾ *nhauṇā* *'to bath'*	ਨ੍ਹਾ *nhā*	ਨ੍ਹਾਉਨਾ *nhāunā*	ਨ੍ਹਾਤਾ *nhātā*	ਨ੍ਹਾਊਂਗਾ *nhāūṅgā*
4.	ਭੌਣਾ *bhauṇā* *'to wonder'*	ਭੌਂ *bhauṃ*	ਭੌਂਨਾਂ *bhaunnāṃ*	ਭੌਂਵਿਆਂ *bhaṃviāṃ*	ਭੌਊਂਗਾ *bhauūṅgā*
5.	ਸੌਣਾ *sauṇā* *'to sleep'*	ਸੌਂ *sauṃ*	ਸੌਂਨਾ *saunnā*	ਸੌਂਵਿਆਂ *saṃviāṃ*	ਸੌਊਂਗਾ *sauṃūṅgā*
6.	ਕਰੌਣਾ *karauṇā* *'to cause to do'*	ਕਰਾ *karā*	ਕਰੌਨਾ *karaunā*	ਕਰਾਇਆ *karāiā*	ਕਰਾਊਂਗਾ *karāūṅgā*
7.	ਲੈਣਾਂ *laiṇāṃ*	ਲੈ *lai*	ਲੈਨਾ *lainā*	ਲਿਆ *liā*	ਲਊਂਗਾ *laūṅgā*

	‘to take’				
8.	ਰਹਿਣਾ *rahiṇā* *‘to live’*	ਰਹਿ *rahi*	ਰਹਿਨਾਂ *rahināṃ*	ਰਿਹਾ *rihā*	ਰਹੂੰਗਾਂ *rahuṅgāṃ*
9.	ਢਹਿਣਾਂ *ḍhahiṇāṃ* *‘to fall’*	ਢਹਿ *ḍhahi*	ਢਹਿਨਾ *ḍhahinā*	ਢਿਹਾ *ḍhihā*	ਢਹੂੰਗਾਂ *ḍhahūṅgāṃ*
10.	ਪੀਣਾ *pīṇā* *‘to drink’*	ਪੀ *pī*	ਪੀਨਾ *pīnā*	ਪੀਤਾ *pītā*	ਪੀਊਂਗਾ *pīūṅgā*
11.	ਪੀਹਣਾ *pīhṇā* *‘to grind’*	ਪੀਹ *pīh*	ਪੀਹਨਾ *pīhnā*	ਪੀਹਾ *pīhā*	ਪੀਹੂੰਗਾ *pīhūṅgā*
12.	ਸਿਉਣਾ *siuṇā* *‘to sew’*	ਸਿਉ *siu*	ਸਿਉਣਾ *siuṇā*	ਸੀਤਾ *sītā*	ਸਿਊਂਗਾ *siūṅgā*
13.	ਕੁੱਟਣਾ *kuṭṭṇā* *‘to beat’*	ਕੁੱਟ *kuṭṭ*	ਕੁੱਟਨਾ *kuṭṭnā*	ਕੁੱਟਿਆ *kuṭṭiā*	ਕੁੱਟੂੰਗਾ *kuṭṭūṅgā*
14.	ਬੰਨ੍ਹਣਾ *bannhṇā* *‘to bind’*	ਬੰਨ੍ਹ *bannh*	ਬੰਨ੍ਹਨਾ *bannhnā*	ਬੰਨ੍ਹਿਆ *bannhiā*	ਬੰਨ੍ਹੂੰਗਾ *bannūṅgā*
15.	ਬੋਲਣਾ *bōlṇā* *‘to say’*	ਬੋਲ *bōl*	ਬੋਲਨਾ *bōlnā*	ਬੋਲਿਆ *bōliā*	ਬੋਲੂੰਗਾ *bōlūṅgā*
16.	ਜਾਣਾ *jāṇā* *‘to go’*	ਜਾ *jā*	ਜਾਨਾ *jānā*	ਗਿਆ *giā*	ਜਾਊਂਗਾ *jāūṅgā*

17.	ਹੋਣਾ *hōṇā* *‘to be’*	ਹੋ *hō*	ਹੁੰਨਾ *hunnā*	ਹੋਇਆ *hōiā*	ਹੋਊਂਗਾ *hōūṅgā*
18.	ਜਾਨਣਾ *jānṇā* *‘to know’*	ਜਾਣ *jāṇ*	ਜਾਨਨਾ *jānnā*	ਜਾਂਦਿਆਂ *jāndiāṃ*	ਜਾਣੂੰਗਾ *jāṇūṅgā*
19.	ਦੌੜਨਾ *dauṛnā* *‘to run’*	ਦੌੜ *dauṛ*	ਦੌੜਨਾ *dauṛnā*	ਦੌੜਿਆ *dauṛiā*	ਦੌੜਾਗਾਂ *dauṛāgāṃ*
20.	ਹਟਾਨਾ *haṭānā* *‘to remove’*	ਹਟਾ *haṭā*	ਹਟਾਨਾ *haṭānā*	ਹਟਾਇਆ *haṭāiā*	ਹਟਾਵਾਂਗਾ *haṭāvāṅgā*

Appendix D

Rule base for function word insertion

S. No	Function word insertion rule base
1.	agt:null:null:null:ਨੇ:@past:VINT#@progress#@custom#link#jA:PRON#3SG:null
2.	agt:null:null:null:ਨੇ:@past:VINT#@progress#kha#@custom#link#jA:N#3SG:null
3.	agt:null:null:null:ਨੇ:@past#@complete:jA#@progress#kha:N#ANIMT:null
4.	agt:null:null:null:ਨੇ:@present#@complete:jA#@progress#kha:N#ANIMT:INANI
5.	agt:null:null:null:ਨੇ:@past#link:jA#@progress#@repeat#@custom#@progress#@ability#@repeat:N#ANIMT:null
6.	agt:null:null:null:ਨੇ:@past#link:Ja#@progress#@custom#@progress#@ability#@repeat:N#MACH:null
7.	agt:null:null:null:ਨੇ:@present#link:jA#@progress#@custom#@progress#@ability#@repeat#@complete:N#MACH:null
8.	agt:null:null:null:ਨੇ:@present#link:jA#@progress#@custom#@progress#@ability#@repeat#@complete:N#ANIMT:null
9.	agt:null:null:null:ਨੇ:@past#ga:null:N#ANIMT:null
10.	agt:null:null:null:ਨੇ:@past#kha:@progress#@custom#@complete:N#ANIMT:null
11.	agt:null:null:null:ਨੇ:@present#kha:@progress#@custom#@complete:N#ANIMT:null
12.	agt:null:null:null:ਨੇ:@future#kha:@progress#@custom#@complete:N#ANIMT:null
13.	agt:null:null:null:ਨੇ:@present#ga:jA#@progress:N#ANIMT:null
14.	agt:null:null:null:ਨੇ:@future#ga:jA#@progress:N#ANIMT:null

15.	and:null:null:ਅਤੇ:null:N:null:N:null
16.	aoj:null:null:null:null:null:null:null:null
17.	bas:null:null:null:ਤੋਂ:CMP:null:N:null
18.	bas:null:null:null:ਤੋਂ:CMP:null:PRON:null
19.	bas:null:null:null:ਤੋਂ:ADJ:null:N:null
20.	bas:null:null:null:ਤੋਂ:ADV:null:N:null
21.	ben:null:null:null:ਦੇ ਲਈ:null:null:N:null
22.	cag:null:null:null:ਦੇ ਨਾਲ:null:null:N:null
23.	cag:null:null:null:ਦੇ ਨਾਲ:null:null:PRON:null
24.	cao:null:null:null:ਦੇ ਨਾਲ:null:null:PRON:null
25.	cao:null:null:null:ਦੇ ਨਾਲ:null:null:N:null
26.	cnt:null:null:null:null:null:null:null:null
27.	cob:null:null:null:ਦੇ ਨਾਲ:null:null:N:null
28.	cob:null:null:null:ਦੇ ਨਾਲ:null:null:PRON:null
29.	coo:ਉਦੋਂ:null:ਜਦੋਂ:null:null:null:null:null
30.	dur:null:null:null:ਦੇ ਵੇਲੇ:null:null:null:null
31.	equ:null:null:null:null:null:null:null:null
32.	fmt:null:ਤੋਂ:null:ਤੱਕ:null:null:N:null
33.	fmt:null:ਤੋਂ:null:ਤੱਕ:null:null:PRON:null
34.	frm:null:null:null:ਦੇ:N#MALE:null:N#PLC:null
35.	frm:null:null:null:ਦੀ:N#FEMALE:null:N#PLC:null
36.	frm:null:null:null:ਤੋਂ:V:null:N:null
37.	frm:null:null:null:ਤੋਂ:null:null:PRON:null
38.	gol:null:null:null:null:null:link:INANI:null
39.	gol:null:null:null:ਵਿਚ:null:null:INANI#N:null
40.	gol:null:null:null:ਵਿਚ:null:null:INANI#PRON:null

41.	gol:null:null:null:ਵਿਚ:null:null:AOJ#N:null
42.	gol:null:null:null:ਨੂੰ:null:null:N#ANI:null
43.	gol:null:null:null:ਨੂੰ:null:null:PRON#ANI:null
44.	icl:null:null:ਤਰਾਂ ਦਾ:null:null:null:null:null
45.	ins:null:null:null:ਨਾਲ:null:null:N:null
46.	ins:null:null:null:ਨਾਲ:null:null:PRON:null
47.	int:null:null:null:null:null:null:null:null
48.	iof:null:null:null:null:null:null:null:null
49.	man:null:null:null:ਨਾਲ:V:ADV:null:TIME#PLC#ADJ#TIME#PLC#ADJ# REFPLC#OFTEN#LETTER
50.	man:null:null:ਬਾਰੇ:null:V:ADV:OFTEN:null
51.	met:null:null:null:ਤੋਂ:null:null:N:null
52.	met:null:null:null:ਤੋਂ:null:null:PRON:null
53.	mod:null:null:null:ਦਾ:@def:null:N:null
54.	mod:null:null:null:ਦਾ:null:MACH:PRON:MOD
55.	mod:null:null:null:ਦੇ ਨਾਲ:@topic:null:N:null
56.	mod:null:null:null:ਦੇ ਨਾਲ:@topic:null:PRON:null
57.	mod:null:null:null:null:null:null:null:null
58.	nam:null:null:null:null:null:null:null:null
59.	obj:null:null:null:ਦਾ:V#da:null:N#ANIMT#MALE:null
60.	obj:null:null:null:ਨੂੰ:V:da#lnk#@female#@future#VOCCUR#VBAN#FST# @complete#NONOON#Vo#@female#link:N:PLC#TIME#NONOON# NOON
61.	obj:null:null:null:ਨੂੰ:CV:null:N#ANIMT:PLC
62.	obj:null:null:null:ਨੂੰ:V#@obligation#@future:null:N#ANIMT:null
63.	obj:null:null:null:ਨੂੰ:V:null:N#NOON:null
64.	obj:null:null:null:ਨੂੰ:V#kha:null:N#ANIMT:null

65.	obj:null:null:null:ਨੂੰ:ADV:da#lnk:N:null
66.	obj:null:null:null:ਵੱਲ:V#VSEE:null:N:null
67.	obj:null:null:null:ਦੀ:V#F:null:N:null
68.	obj:null:null:null:ਵਿਚ:CV:NOACT:N#PLC:null
69.	obj:null:null:null:ਵਿਚ:VOCCUR:null:TIME:null
70.	opl:null:null:null:ਵਿਚ:null:null:N:null
71.	opl:null:null:null:ਵਿਚ:null:null:PRON:null
72.	per:null:null:null:ਵਾਰ:null:null:null:null
73.	plc:null:null:null:ਵਿਚ:null:jA#NOACT:N:null
74.	plc:null:null:null:ਵਿਚ:null:jA#NOACT:PRON:null
75.	plf:null:null:null:ਤੋਂ:null:null:null:null
76.	plt:null:null:null:ਤੱਕ:null:jA:null:null
77.	pof:null:null:null:ਦੇ:MALE#@pl:null:N:null
78.	pof:null:null:null:ਦਾ:MALE:@pl:N:@pl
79.	pof:null:null:null:ਦੀ:FEMALE:@pl:N:@pl
80.	pof:null:null:null:ਦੀਆਂ:FEMALE#@pl:null:N:null
81.	pof:null:null:null:ਦੇ:MALE#@pl:null:PRON:null
82.	pof:null:null:null:ਦਾ:MALE:@pl:PRON:@pl
83.	pof:null:null:null:ਦੀਆਂ:FEMALE#@pl:null:PRON:null
84.	pos:null:null:null:ਦਾ:null:null:null:1SG#PRON
85.	ptn:null:null:null:ਦੇ ਨਾਲ:null:null:N:null
86.	ptn:null:null:null:ਦੇ ਨਾਲ:null:null:PRON:null
87.	pur:null:null:null:ਦੇ ਲਈ:null:null:N:null
88.	pur:null:null:null:ਦੇ ਲਈ:null:null:PRON:null
89.	pur:null:null:null:ਦੇ ਲਈ:null:null:V:null
90.	qua:null:null:null:null:null:null:null:null

91.	rsn:null:null:null:ਕਾਰਨ:null:null:null:N
92.	rsn:ਦਾ ਕਾਰਨ :null:null:null:null:jA:N#MACH#@pl:null
93.	rsn:ਦੇ ਕਾਰਨ :null:null:null:null:jA:N#MACH:null
94.	rsn:ਕਾਰਨ :null:null:null:jA:null:N:null
95.	rsn:ਦੇ ਕਾਰਨ :null:null:null:null:jA:N#@pl:null
96.	scn:null:null:null:ਵਿਚ:null:null:N:null
97.	scn:null:null:null:ਵਿਚ:null:null:PRON:null
98.	seq:null:null:null:ਦੇ ਬਾਅਦ:null:null:N:null
99.	seq:null:null:null:ਦੇ ਬਾਅਦ:null:null:PRON:null
100.	src:null:null:null:ਤੋਂ:null:null:N:null
101.	src:null:null:null:ਤੋਂ:null:null:PRON:null
102.	tim:null:null:null:ਵਿਚ:nll:null:N:null
103.	tim:null:null:null:ਵਿਚ:null:null:PRON:null
104.	tim:null:null:null:ਵਿਚ:VOCCUR:null:TIME:null
105.	tim:null:null:null:ਨੂੰ:V:VOCCUR:TIME:null
106.	tim:null:null:null:ਨੂੰ:CV:V:TIME:null
107.	tim:null:null:null:ਨੂੰ:N:null:TIME:null
108.	tmf:null:null:null:ਤੋਂ:null:null:N:null
109.	tmf:null:null:null:ਤੋਂ:null:null:PRON:null
110.	tmt:null:null:null:ਤੱਕ:null:null:N:null
111.	tmt:null:null:null:ਤੱਕ:null:null:PRON:null
112.	to:null:null:null:ਨੂੰ:N:null:PLC:null
113.	to:null:null:null:ਦੇ ਨਾਲ:null:null:N:PLC
114.	to:null:null:null:ਦੇ ਨਾਲ:null:null:PRON:PLC
115.	via:null:null:null:ਵੱਲੋਂ:null:null:N:null
116.	via:null:null:null:ਵੱਲੋਂ:null:null:PRON:null

List of Publications by the Author

Papers in refereed journals

1. Parteek Kumar, R K Sharma, Generation of UNL Attributes and Resolving Relations for Punjabi Enconverter. *Malaysian Journal of Computer Science (ISSN 0127-9084), Vol.24, No.1, March 2011: 34-46.*
2. Parteek Bhatia, R K Sharma, Generation of Case Markers for UNL-Punjabi Deconverter. *International Journal of Systemics, Cybernetics and Informatics (ISSN 0973-4864), April 2009: 44-48.*
3. Parteek Bhatia, R K Sharma, Universal Networking Language for Language Independent Net. *CSI Communication, Vol.31, No.9, December 2008: 38-42.*
4. Parteek Kumar, R K Sharma, Punjabi to UNL EnConversion System. *SADHANA Academy Proceedings in Engineering Sciences, Accepted.*

Papers in conference proceedings

1. Parteek Bhatia, R K Sharma, Designing of Rule Base for Punjabi-UNL Enconverter. *Proc. IEEE Int. Conf. on Advanced Computing*, Thapar University, Patiala, March 2009, 3763-3768.
2. Parteek Bhatia, R K Sharma, Role of Punjabi Morphology in Designing of Punjabi-UNL Enconverter. *Proc. ACM Int. Conf. on Computing, Communication and Control*, Mumbai, India, January 2009, 562-566.

References

1. Adly N, Alansary S 2009 Evaluation of Arabic machine translation system based on the Universal Networking Language. *Natural Language Processing and Information Systems, Lecture Notes in Computer Science,* 5723: 243-257.
2. Alansary S, Nagi M, Adly N 2006 Towards a language-independent Universal Digital Library. *Proc. 2nd Int. Conf. on Universal Digital Library,* Alexandria, Egypt, 1-10.
3. Alansary S, Nagi M, Adly N 2007 A Semantic-based approach for multilingual translation of massive documents. *Proc. 7th Int. Symposium on Natural Language Processing,* Chonburi, Thailand, 317-323.
4. Ali Y, Das J, Abdullah S M, Nurannabi A M 2008 Morphological analysis of Bangla words for Universal Networking Language. *Proc. 3rd Int. Conf. on Digital Information Management,* London, England, 532-537.
5. Anthes G 2010 Automated translation of Indian languages. *Communications of the ACM,* 53 (1): 24-26.
6. Aref M, Al-Mulhem M, Al-Muhtaseb H 1995 English to Arabic Machine Translation: a critical review and suggestions for development. *Proc. 4th Saudi Engineering Conference,* Saudi Arabia, 421-427.
7. Arnold D, Balkan L, Meijer S, Humphreys R L, Sadler L 1994 Machine Translation: an Introductory Guide. *NCC Blackwell,* London.
8. Avetisyan A, Avetisyan V 2010 LOOK4: Enhancement of web search results with Universal Words and WordNet. *Proc. 5th Int. Conf. on Global WordNet,* Mumbai, India, 1-5.
9. Bahri U S, Walia P S 2003 Introductory Punjabi. *Publication Bureu, Punjabi University,* Patiala.
10. Balajapally P, Bandaru P, Ganapathiraju M, Balakrishnan N, Reedy R 2006 Multilingual book reader: transliteration, word-to-word translation and full-text translation. *Proc. 13th Biennial Conf. of Victorian Association for Library Automation,* Melbourne, Australia, 1-12.

11. Balkan L 1998 Test suites: some issues in their use and design. *Proc. Int. Conf. on Machine Translation: ten years on*, England, 214-222.
12. Bekios J, Boguslavsky I, Cardeñosa J, Gallardo C 2007 Using Wordnet for building an interlingual dictionary. *Proc. 5th Int. Conf. Information Research and Applications*, Varna, Bulgaria, 1-8.
13. Berger A, Brown P, Pietra S, Pietra V, GiUett J, Lafferty J, Mercer R, Printz H, Urei L 1994 The Candide system for Machine Translation. *Proc. Int. Conf. on Human Language Technology*, USA, 157-162.
14. Bértoli L, Luz R P, Bastos R C 2005 A web platform using UNL: CELTA's showcase. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E*, México, Research on Computing Science: 276-285.
15. Bharati A, Chaitanya V, Sangal R 1994 Natural Language Processing: A Paninian Perspective. *Prentice-Hall of India*, New Delhi.
16. Bharati A, Misra D, Bai L, Sangal R 2006 AnnCorra: Annotating corpora guidelines for POS and chunk annotation for Indian languages. *Technical Report, Language Technologies Research Centre*, IIIT, Hyderabad, India: 1-38.
17. Bharati A, Sangal R 1993 Parsing free word order languages in the Paninian framework. *Proc. Annual Meeting of Association for Computational Linguistics*, New Jersey, 105-111.
18. Bharati A, Sangal R, Sharma D 2007 SSF: Shakti Standard Format Guide. *Technical Report, Language Technologies Research Centre, International Institute of Information Technology*, Hyderabad, India: 1-25.
19. Bhattacharyya P 2001 Multilingual information processing through Universal Networking Language. *Indo UK Workshop on Language Engineering for South Asian Languages*, Mumbai, India, 1-10.
20. Blanc É 2005 About and around the French Enconverter and the French Deconverter. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 157-166.

21. Boguslavsky I M, Iomdin L, Sizov V G 2005 Interactive EnConversion by means of the ETAP-3 system. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 230-240.
22. Boitet C, Bhattacharyya P, Blanc E, Meena S, Boudhh S, Fafiotte G, Falaise A, Vachhani V 2007 Building Hindi-French-English-UNL resources for SurviTra-CIFLI, a linguistic survival system under construction. *Proc. 7th Int. Symp. on Natural Language Processing,* Chonburi, Thailand, 1-6.
23. Borra A, Chan E, Lim C, Tan R, Tong M 2007 LFG-based machine translation engine for English and Filipino. *Proc. 4th National Natural Language Processing, Research Symposium Philippine Languages and Computation,* Manila, 36-42.
24. Boudhh S, Bhattacharyya P 2009 Unification of Universal Word dictionaries Using WordNet ontology and similarity measures. *Proc. 7th Int. Conf. on Computer Science and Information Technologies,* Armenia, 271-283.
25. Brockett C, Aikawa T, Aue A, Menezes A, Quirk C, Suzuki H 2002 English-Japanese example-based Machine Translation using abstract linguistic representations. *Proc. of Int. Conf. Computational Linguistics*, Taiwan, 1-7.
26. Brown P F, Cocke J, Pietra S, Pietra V, Jelinek F, Lafferty J, Mercer R, Roossin P 1990 A statistical approach to machine translation. *Computational Linguistics*, 16(2): 79–85.
27. Bueno T, Hoeschl H, Bortolon A, Mattos E, Santos C, Barcia R 2005 Knowledge engineering suite: a tool to create ontologies for automatic knowledge representation in intelligent systems. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 337-346.
28. Cardeñosa J, Gallardo C, Tovar E 2005a Standardization of the generation process in a multilingual environment. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 10-24.
29. Cardeñosa J, Gallardo C, Iraola L 2005b An XML-UNL model for knowledge-based annotation. *Universal Network Language: Advances in Theory and*

Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E, México, Research on Computing Science: 300-308.

30. Cardeñosa J, Gelbukh A, Tovar E (Eds.) 2005c *Universal Network Language: Advances in Theory and Applications*, México, Research on Computing Science: 1-466.
31. Chakrabarti D, Sarma V, Bhattacharyya P 2006 Hindi verb knowledge base and noun incorporation in Hindi. *Proc. 3rd Global Wordnet Conference*, Korea, 43-50.
32. Chaudhury S, Rao A, Sharma D M 2010 Anusaaraka: An expert system based Machine Translation system. *Proc. Int. Conf. on Natural Language Processing and Knowledge Engineering*, Beijing, China, 1-6.
33. Choudhary B, Bhattacharyya P 2002 Text clustering using Universal Networking Language representation. *Proc. 11th Int. Conf. on World Wide Web*, Hawaii, USA, 1-7.
34. Choudhury M, Ershadul H, Nawab Y A, Mohammad Z H S, Ahsan R M 2005 Bridging Bangla to Universal Networking Language- a human language neutral meta-language. *Proc. 8th Int. Conf. on Computer and Information Technology*, Dhaka, Bangladesh, 104-109.
35. Daoud D M 2005 Arabic Generation in the Framework of the Universal Networking Language. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 195-209.
36. Dave S, Bhattacharyya P 2001 Knowledge extraction from Hindi text. *J. of Institution of Electronic and Telecommunication Engineers*, 18 (4): 1-12.
37. Dave S, Parikh J, Bhattacharyya P 2001 Interlingua based English Hindi machine translation and language divergence. *J. of Machine Translation (JMT),* 16(4): 251-304.
38. Dey K, Bhattacharyya P 2005 Universal Networking Language based analysis and generation of Bengali case structure constructs. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 215-229.

39. Dhanabalan T, Geetha T V 2003 UNL DeConverter for Tamil. *Proc. Int. Conf. on the Convergence of Knowledge, Culture, Language and Information Technologies,* Alexandria, Egypt, 1-6.
40. Dhanabalan T, Saravanan K, Geetha T V 2002 Tamil to UNL EnConverter. *Proc. Int. Conf. on Universal Knowledge and Language,* Goa, India, 1-16.
41. Dorr J B 1987 UNITRAN: an interlingual machine translation system. *Technical Report, Massachusetts Institute of Technology*, Cambridge: 1-12.
42. Dwivedi S K, Sukhadeve P 2010 Machine Translation System in Indian perspectives. *J. of Computer Science,* 6(10): 1111-1116.
43. Dwivedi V 2002 Generation of Hindi from Universal Networking Language. M. Tech. Thesis, IIT Bombay, Mumbai.
44. Farghaly A, Senellart J 2003 Intuitive coding of the Arabic lexicon. *Proc. 9th MT Summit*, USA, 1-5.
45. Fat G 2005 T2CMT: Tagalog-to-Cebuano machine translation. *Proc. 5th Philippine Computing Science Congress*, Cebu City, Philippines, 21-23.
46. Gill H S, Gleason H 1986 A reference grammar of Punjabi. *Publication Bureu, Punjabi University*, Patiala.
47. Gill M S 2008, Development of a Punjabi grammar checker. Ph.D. Thesis, Punjabi University, Patiala.
48. Giri L 2000 Semantic net like knowledge structure generation from natural languages. M. Tech. Thesis, IIT Bombay, India.
49. Gore L, Patil N 2002 English to Hindi - Translation System. *Proc. Symposium on Translation Support Systems*, IIT Kanpur, India, 178-184.
50. Goyal V 2010 Development of a Hindi to Punjabi Machine Translation system. Ph.D. Thesis, Punjabi University, Patiala.
51. Goyal V, Lehal G S 2010 Web based Hindi to Punjabi Machine Translation system. *J. of Emerging Technologies in Web Intelligence,* 2(2): 148-151.
52. Hajlaoui N, Boitet C 2005 A Pivot XML-based architecture for multilingual, multiversion documents: parallel monolingual documents aligned through a central correspondence descriptor and possible use of UNL. *Universal Network*

Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E, México, Research on Computing Science: 309-325.

53. Hrushikesh B 2002 Towards Marathi sentence generation from Universal Networking Language. M. Tech. Thesis, IIT Bombay, Mumbai.
54. Hutchins W J 1995 Machine Translation: a brief history. *Proc. Concise history of the Language Sciences: From the sumerians to the cognitivists,* Pergamon, 431-445.
55. Hutchins W J 1997 From first conception to first demonstration: the nascent years of Machine Translation, 1947-1954, a chronology. *J. of Machine Translation*, 12(3): 195-252.
56. Hutchins W J 2003 Machine translation: half a century of research and use. *Proc. UNED summer school,* Ávila, Spain, 1-24.
57. Hutchins W J, Somers H L 1992 An introduction to Machine Translation. London, *Academic Press*.
58. Iraola L 2005 Using WordNet for linking UWs to the UNL UW system. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 369-378.
59. Iyer J A, Bhattacharyya P 2005 Using semantic information to improve case retrieval in case-based reasoning systems. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E*, México, Research on Computing Science: 347-358.
60. Jain K 2005 UNL to Hindi generation lexicon and morphology. M. Tech. Thesis, IIT Bombay, Mumbai.
61. Jain M, Damani O P 2009 English to UNL (Interlingua) EnConversion. *Proc. 2nd Conference on Language and Technology,* Lahore, Pakistan, 1-8.
62. Jain R, Sinha R M K, Jai A 1995 Role of Examples in Translation. *Proc. Int. Conf. on Systems, Man and Cybernetics,* Vancouver, Canada, 1615-1620.
63. Jiang C, Tissiani G, Falquet G, Rodolfo P 2005 Facilitating communication between languages and cultures: a computerized interface and knowledge base. *Universal Network Language: Advances in Theory and Applications, Ed(s)*

Cardeñosa J, Gelbukh A, Tovar E, México, Research on Computing Science: 358-368.

64. Josan G S, Lehal G S 2008 A Punjabi to Hindi Machine Translation system. *Proc. 22nd Int. Conf. on Computational Linguistics*, Manchester, 157-160.
65. Jurafsky D, Martin J H 2000 Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. *Prentice Hall*, USA.
66. Karande J B 2007 Multilingual search engine: implementation using UNL. *Proc. Int. Conf. on Semantic Web and Digital Libraries*, Bangalore, India, 1-7.
67. Keshari B, Bista S K 2005 UNL Nepali Deconverter. *Proc. 3rd International CALIBER*, Cochin, 70-76.
68. Koehn P, Birch A, Steinberger R 2009 462 Machine Translation systems for Europe. *Proc. 9th MT Summit*, Louisiana, USA, 1-8.
69. Kulkarni A P 2003 Design and architecture of Anusaaraka: An approach to Machine Translation. *Satyam Technology Review*, 1(4): 57-64.
70. Lafourcade M 2005 Semantic analysis through ant algorithms, conceptual vectors and fuzzy UNL graphs. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 125-137.
71. Lafourcade M, Boitet C 2002 UNL Lexical selection with conceptual vectors. *Proc. Int. Conf. on Language Resources and Evaluation*, Spain, 1958-1964.
72. Lagarda A, Casacuberta F 2008 Applying boosting to statistical machine translation. *Proc. Int. Conf. of EAMT*, Hamburg, Germany, 88-96.
73. LDC 2005 Linguistic data annotation specification: assessment of adequacy and fluency in translations. *Revision 1.5, Technical Report, Linguistic Data Consortium.*
74. Lepage Y, Denoual E 2005 ALEPH: an EBMT system based on the preservation of proportional analogies between sentences across languages. *Proc. Int. Workshop on Spoken Language Translation: Evaluation Campaign on Spoken Language Translation*, Pittsburgh, 1-8.

75. Lewis, Paul M (Eds.) 2009 Ethnologue: Languages of the World. *Sixteenth edition, SIL International*, Dallas.

76. Martins R T, Hasegawa R, Graças M, Nunes V 2005 Hermeto: A NL–UNL Enconverting Environment. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 254-260.

77. Martins R T, Rino L H M, Nunes M G V, Oliveira O N 2002 The UNL distinctive features: evidences through a NL-UNL encoding task. *Proc. 1st Int. Workshop on UNL, other Interlinguas and their Applications*, Las Palmas, Spain, 8-13.

78. Martins R T, Rino L H M, Osvaldo O N, Hasegawa R, Nunes M G V 1997 Specification of the UNL-Portuguese enconverter-deconverter prototype. *Relatórios Técnicos do ICMC-USP*, 1-10.

79. Mitamura T, Nyberg E H 1995 Controlled English for knowledge-based MT: experience with the KANT system. *Proc. 6th Int. Conf. on Theoretical and Methodological Issues in Machine Translation* (*TMI)*, Belgium, 158-172.

80. Mitamura T, Nyberg E H 2000 The KANTOO machine translation environment. *Proc. Conf. of the Association for Machine Translation in the Americas (AMTA), (Ed.) White J S, LNAI*, 1934: 192-195.

81. Mohanty R, Almeida A, Bhattacharyya P 2005a Prepositional phrase attachment and interlingua. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 241-253.

82. Mohanty R, Dutta A, Bhattacharyya P 2005b Semantically relatable sets: building blocks for representing semantics. *Proc. 10th MT Summit*, Phuket, 1-8.

83. Montesco C E, Moreira D A 2005 UCL-universal communication language. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 326-336.

84. Mridha M F, Hossain M Z, Noor S A 2010 Development of Morphological Rules for Bangla Words for Universal Networking Language. *Int. J. of Computer Science and Network Security,* 10 (10): 231-237.

85. Mukerjee A, Achla M R, Kumar K, Goyal P, Shukla P 2005 Universal Networking Language: a tool for language independent semantics? *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 145-154.
86. Nagi M, Senta T D, 2008 Sharing knowledge across language barriers: a universal approach for online books. *Proc. ACM workshop on Research advances in large digital book repositories,* California, USA, 49-52.
87. Nalawade A 2007 Natural language generation from Universal Networking Language. M. Tech. Thesis, IIT Bombay, Mumbai.
88. Naskar S K, Bandyopadhyay S 2005 Use of Machine Translation in India: current status. *Proc. of 10th MT Summit,* Thailand: 465-470.
89. Naskar S K, Bandyopadhyay S 2006 A Phrasal EBMT system for translating English to Bengali. *Proc. Workshop on Language, Artificial Intelligence, and Computer Science for Natural Language Processing Applications*, Bangkok, Thailand, 373-379.
90. Navarro J, Gonzalez J, Pico D, Casacuberta F, Val J, Fabregat F, Pla F, Tomas J 2004 SisHiTra: a Spanish-to-Catalan hybrid machine translation system. *Lecture Notes in Computer Science*, 3230: 349–359.
91. Nguyen P T, Ishizuka M 2006 A statistical approach for Universal Networking Language-based relation extraction. *Proc. Int. Conf. on Research, Innovation and Vision for the Future*, Ho Chi Minh City, Vietnam, 153-160.
92. Pelizzoni J M, Nunes M V 2005 Flexibility, configurability and optimality in UNL deconversion via multiparadigm programming. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 175-194.
93. Ramamritham K, Bahuman A, Duttagupta S 2006 aAqua: a database-backended multilingual, multimedia community forum. *Proc. Int. Conf. on Management of Data*, Chicago, USA: 784-786.
94. Ramanathan A 2008 Statistical machine translation, Ph.D. Seminar Report, IIT Bombay, Mumbai.

95. Ramanathan A, Choudhary H, Ghosh A, Bhattacharyya P 2009 Case markers and morphology: addressing the crux of the fluency problem in English-Hindi SMT. *Proc. Joint Conference of the 47th Annual Meeting of the ACL and 4th International Joint Conference on Natural Language Processing*, Suntec, Singapore, 800–808.
96. Ramanathan A, Kavitha M, Hegde J, Shekhar C, Shah R, Bade S, Sasikumar M 2006 MaTra: A practical approach to fully automatic indicative English-Hindi Machine Translation. *Proc. 1st Nat. Symposium on Modeling and Shallow Parsing of Indian Languages*, Mumbai, India, 1-8.
97. Ribeiro C, Santos R, Chaves R P, Marrafa P 2004 Semi-automatic UNL dictionary generation using WordNet.PT. *Proc. 4th Int. Conf. on Language Resources and Evaluation*, Lisbon, 279-282.
98. Rouquet D, Nguyen H 2009 Interlingual annotation of texts in the OMNIA project. *Proc. 4th Int. Conf. on Language and Technology*, Poland, 1-5.
99. Saha G K 2005 The EB-ANUBAD translator: a hybrid scheme. *J. of Zhejiang University Science*, 6A(10):1047-1050.
100. Sangal R, Sharma D M 2004 Creating language resources for NLP in Indian languages. *Proc. Int. Conf. on Crossing the Digital Divide Shaping Technologies to Meet Human Needs*, Kathmandu, Nepal, 1-13.
101. Schwenk H, Rauf S A, Barrault L 2009 SMT and SPE machine translation systems for WMT'09. *Proc. 4th EACL Workshop on Statistical Machine Translation*, Athens, Greece, 130–134.
102. Scott B, Barreiro A 2009 OpenLogos MT and the SAL representation language. *Proc. 1st Int. Workshop on Free/Open-Source Rule-Based Machine Translation*, Alicante, Spain, 19–26.
103. Seasly J 2003 Machine Translation: a survey of approaches. *Technical Report, University of Michigan*, Ann Arbor.
104. Senellart J, Dienes P, Váradi T 2001 New generation SYSTRAN translation system. *Proc. 8th MT Summit*, Spain, 1-6.
105. Sérasset G, Boitet C 2000 On UNL as the future "html of the linguistic content" & the reuse of existing NLP components in UNL-related applications with the

example of a UNLFrench deconverter. *Proc. Int. Conf. on Computational Linguistics*, Saarbruecken, Germany, 768-774.

106. Shah C, Parikh J, Soni T 2000 A conversion system from English to Universal Networking Language. B.E Dissertation, Dharamsinh Desai Institute of Technology, Nadiad.
107. Shi X, Chen Y 2005 A UNL Deconverter for Chinese Universal Network Language. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 167-174.
108. Shukla P, Mukherjee A, Raina A 2004 Towards a language independent encoding of documents a novel approach to multilingual question answering. *Proc. 1st Int. Workshop on Natural Language Understanding and Cognitive Science*, Porto, Portugal: 116-125.
109. Siddiqui T, Tiwary U S 2008 Natural language processing and information retrieval. *Oxford University Press*, New Delhi.
110. Singh H, Singh G L 1980 ਕਾਲਜ ਪੰਜਾਬੀ ਵਿਆਕਰਨ *kālaj pañjābī viākaran*. *Punjab State University Text- Book Board*, Chandigarh.
111. Singh S 2007 Parsing and syntax planning for UNL Punjabi DeConverter. M.E. Thesis, Thapar University, Patiala.
112. Singh S, Dalal M, Vachhani V, Bhattacharyya P, Damani O P 2007 Hindi generation from interlingua. *Proc. 11th MT Summit*, Copenhagen, Denmark 1-8.
113. Singh T D, Bandyopadhyay S 2010 Manipuri-English bidirectional statistical Machine Translation systems using morphology and dependency relations. *Proc. 23rd Int. Conf. on Computational Linguistics*, Beijing, 75-83.
114. Sinha R 2005a Hindi generation: Syntax planning and case marking. Mini Project Report, IIT Bombay, Mumbai.
115. Sinha R M K 1984 Computer processing of Indian languages and scripts - potentialities and problems. *J. of Institution of Electronic and Telecommunication Engineers*, 30(6):133-149.

116. Sinha R M K 2005b Integrating CAT and MT in AnglaBharti-II architecture. *Proc. European Association for Machine Translation (EAMT) Conf.*, Budapest, Hungary, 1-10.
117. Sinha R M K 2009 Developing English-Urdu machine translation via Hindi. *Proc. 3rd Workshop on Computational Approaches to Arabic Script-based Languages, 12th MT Summit*, Ottawa, Canada, 89-95.
118. Sinha R M K, Jain A 2003 AnglaHindi: an English to Hindi machine translation system. *Proc. 9th MT Summit*, New Orleans, USA, 1-5.
119. Sinha R M K, Sivaraman K, Agrawal A, Jain R, Srivastava R, Jain A 1995 ANGLABHARTI: A multilingual machine aided translation project on translation from English to Indian languages. *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, Vancouver, Canada, 1609-1614.
120. Sinha R M K, Thakur A 2005 Machine Translation of bi-lingual Hindi-English (Hinglish) Text. *Proc. 10th MT summit*, Phuket, Thailand, 1-8.
121. Slocum J 1984 Machine Translation: its history, current status, and future prospects. *Proc. 10th Int. Conf. on Computational Linguistics*, California, 546-561.
122. Slype V 1979 Critical Study of methods for evaluating the quality of Machine Translation. *Technical Report, European Commission*, Brussels: 1-185.
123. Somal K S, Singh J, Shergill S 2005 ਅਜੋਕੀ ਪੰਜਾਬੀ ਦਾ ਵਿਆਕਰਨ ਅਤੇ ਲੇਖ-ਰਚਨਾ *ajōkī pañjābī dā viākaran atē lēkh-racnā*. *Punjab School Education Board*, Mohali.
124. Sornlertlamvanich V, Potipiti T, Charoenporn T 2000 Thai lexical semantic annotation by UW. *Proc. 7th Int. Workshop on Academic Information Networks and Systems*, Bangkok, Thailand: 1-6.
125. Spanish Language Center 2004. http://www.unl.fi.upm.es/english/index.htm.
126. Srikanth R P, Kapoor D 2001 Machine translation set for quantum leap in India. *Express Computer*.
127. Subramanian R, Narayanan A 1990 An AI-based approach to machine translation in Indian languages. *Communications of the ACM*, 33(5): 521-527.

128. Sumita E, Iida H, Kohyama H 1990 Translating with examples: a new approach to Machine Translation. *Proc. 3rd Int. Conf. on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Texas, 203-212.

129. Surve M, Singh S, Kagathara S, Venkatasivaramasastry K, Dubey S, Rane G, Saraswati J, Badodekar S, Iyer A, Almeida A, Nikam R, Perez C G, Bhattacharyya P 2004 AgroExplorer: a meaning based multilingual search engine. *Proc. Int. Conf. on Digital Libraries*, Delhi, India: 1-13.

130. Tanaka H, Ishizaki S, Uehara A, Uchida H 1989 Research and development of cooperation project on a machine translation system for Japan and its neighboring countries. *Proc. of 2nd MT Summit*, Munich, 146-151.

131. Tomokiyo M, Chollet G 2003 VoiceUNL : a proposal to represent speech control mechanisms within the Universal Networking Digital Language. *Proc. Int. Conf. on the Convergence of Knowledge, Culture, Language and Information Technologies,* Alexandria, Egypt, 1-6.

132. Uchida H 1987 ATLAS: Fujitsu Machine Translation system. *Proc. MT Summit*, Japan, 129-134.

133. Uchida H 1989a ATLAS II: A Machine translation system using conceptual structure as an interlingua. *Machine Translation Summit, (Eds) Nagao M, Tanaka H, Makino T, Nomura H, Uchida H, Ishizaki S*, Ohmsha, Japan: 93-100.

134. Uchida H 1989b Interlingua: Necessity of interlingua for multilingual translation. *Machine Translation Summit, (Eds) Nagao M, Tanaka H, Makino T, Nomura H, Uchida H, Ishizaki S*, Ohmsha, Japan: 29.

135. Uchida H 2005 Universal Networking Language (UNL): Specifications version 2005, *UNDL Foundation*.

136. Uchida H, Sugiyama K 1980 A Machine Translation system from Japanese into English based on conceptual structure. *Proc. 8th Conf. on Computational Linguistics*, Tokyo, Japan, 455-462.

137. Uchida H, Zhu M 1993 Interlingua for multilingual machine translation. *Proc. 4th MT Summit*, Japan, 157-169.

138. Uchida H, Zhu M 2001 The Universal Networking Language beyond Machine Translation. *Proc. Int. Symposium on Language in Cyberspace*, Seoul, Korea, 1-15.
139. Uchida H, Zhu M 2003 Universal Parser. *UNL Center, UNDL Foundation*.
140. Uchida H, Zhu M, 2005 UNL2005 for providing knowledge infrastructure. *Proc. Semantic Computing Workshop*, Chiba, Japan, 1-12.
141. Uchida H, Zhu M, Senta T D 1999 The UNL, A gift for a millennium. *Institute of Advanced Studies, The United Nations University,* Tokyo, Japan.
142. UNDL Foundation 2010. http://www.undl.org.
143. Universal Networking Language (UNL) Center 2000 EnConverter Specifications. *UNDL Foundation*: 1-33.
144. Vachhani V 2006 UNL to Hindi Deconverter. B.E. Dissertation, Dharamsinh Desai Institute of Technology, Nadiad.
145. Vashee K, Gibbs R 2010 Scenarios for customizing an SMT engine based on availability of data. *Proc. 9th Conf. of the Association for Machine Translation in the Americas*, Denver, Colorado, 1-4.
146. Verma N, Bhattacharyya P 2005 Automatic generation of multilingual lexicon by using Wordnet. *Universal Network Language: Advances in Theory and Applications, Ed(s) Cardeñosa J, Gelbukh A, Tovar E,* México, Research on Computing Science: 380-391.
147. Vora A 2002 Generation of Hindi sentences from Universal Networking Language. B.E. Dissertation, Dharamsinh Desai Institute of Technology, Nadiad.
148. Witkam T 1988 DLT: an industrial R & D project for multilingual MT. *Proc. 12th Conf. on Computational Linguistics*, Budapest, 757-759.
149. Yurtseven L 1997 LOGOS machine translation system. *Proc. of 6th MT Summit*, California, USA, 251-252.